VIRTUAL INSTRUMENTATION BASED FUZZY LOGIC CONTROL FOR LIQUID LEVEL

A Thesis submitted
in partial fulfilment of the requirements
for the Degree of

Master of Technology

by

Mahesh R. Ghivari

to the

Department of Chemical Engineering

INDIAN INSTITUTE OF TECHNOLOGY, KANPUR

February 2001

27-2-2001

CERTIFICATE

This is to certify that the work contained in the thesis entitled "**VIRTUAL INSTRUMENTATION BASED FUZZY LOGIC CONTROL FOR LIQUID LEVEL**" has been carried out by ***Mahesh R. Ghivari*** under my supervision and that it has not been submitted elsewhere for a degree.

Ashok Khanna

February 2001

Dr. Ashok Khanna
Professor,
Dept. of Chemical Engineering,
Indian Institute of Technology,
Kanpur – 208 016, India

Acknowledgements

To my thesis supervisor Dr. Ashok Khanna, for the valuable advice and encouragement that I got from him throughout. My gratitude to him is immense and cannot really be expressed in words. I consider it a rare privilege that I could work under him.

To all the professors of the department who showed the rigors and beauties of Chemical Engineering and encouraged me in all ways.

To Dr. Joseph John, Dr. Kamal Poddar, Dr. Sanjay Gupta who helped me to get out of the hardware and VI problems. The AE698a course is greatly acknowledged which wiped out fear of Instrumentation word for a Non-Instrumentation Engineer.

To Dr. P.K. Kalra for assisting me in understanding the concepts of fuzzy control.

To Tarun and Vikas, both BTP students, without whom this thesis would not have been achieved its goal in specified time.

To all my lab mates Debjit, Akhlaq, Rajaneesh, Raj, Tapan, Shubhankar. The wonderful time I had here with my lab mates will be remembered, forever. I am also thankful to Mr. Chauhan and Mr. Ansari for their kind co-operation.

To all my friends – within and outside IIT.

To IITK itself, for its stimulating academic environment. And also for gifting me with many special moments.

To my alma mater T.K.I.E.T., Warnanagar and first job experience at Lote. For those lovely memories that can sail one through all the rough rides in life.

To my family members who kept ringing me up and never gave a feeling of living miles of away from home.

Finally, to all those who helped me in direct or indirect way.

Financial support from MHRD project *Upgradation of process control laboratory* is gratefully acknowledged.

Dedicated

to

My Parents

Abstract

The Virtual Instrumentation is a user-friendly tool in instrumentation work in process control. Software package LabVIEW has been used here for developing virtual instruments (VIs) to be used in process control. A VI for PC-based control of liquid level has been developed. A float type of transducer has been used for measuring liquid level and a potentiometer gives valve position, which, acts as the final control element. For measuring analog input signals; liquid level and valve position and sending analog output signal to actuator PC-interfacing has been done. Virtual Instrument based algorithm for measurement of signals, conversion of these measured signals to engineering units has been developed. Similarly a VI algorithm for constant speed motor control has been developed. Additionally different control algorithms based on process reaction curve tuning, autotuning, gain scheduling and fuzzy logic controller have been implemented and studied. From process reaction curve of liquid level PI-parameters using Cohen-Coon settings have been determined. In autotuning, setpoint relay feedback method, which determines the PI tuning parameters for a PI-control using Zigler-Nichols closed loop procedure have been determined. In Gain Scheduling PI-parameters calculated from autotuning procedure at different operating points, have been obtained. A fuzzy control of the PI type has been developed by understanding dynamics of the liquid level. Using heuristic language *if-then* type of rules have been constructed. These if-then rules are mapped on to membership functions for error, rate of change of error and rate of change of valve position. This fuzzy logic procedure has been evolved in to a 21-rule database. Based on performance index using least squares minimization fuzzy controller gives acceptable set point tracking with least erratic valve movement.

Contents

Chapter

List of Figures

List of Tables

Nomenclature

A_k	k-th fuzzy set
e	error
G	Graphic
h	liquid level
h_{SP}	liquid level set point
k_c	proportional gain
L	linearity factor
P_k	k-th parameter
t	time
t_d	dead time
u	controller output (flow)
x_k	linguistic variable
Δe	change in error
Δh	change in height
Δt	discrete time interval
Δu	change in output (flow change)

Greek letters:

τ_I	integral time
τ_D	derivative time
τ_P	time constant
β	set point factor
β_{ij}	truth value of i-th rule and j-th dataset
μ	membership function, degree of truth

Abbreviations:

AN	Application note
CO	controller Output
Fs	fast
GPIB	General Purpose Instrument Bus

LabVIEW	Laboratory Virtual Instrument Engineers Workbench
Lg	large
MV	manipulated variable
neg.	negative
NI	National Instruments
pos	positive
PV	process variable
PV_f	process variable filtered
Sl	slow
Sm	small
SP	set point
VI	Virtual Instrument
VP	valve position

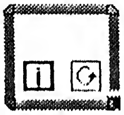
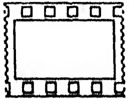
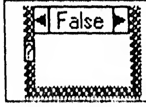
List of symbols of the Functions and Controls used in G-language (LabVIEW) programming:

Function	Block Diagram Function (Numeric)
+	Add
−	Subtract
×	Multiply
÷	Divide
||	Absolute Value
^	Compound Arithmetic
DBL	To double precision float
U32	To unsigned integer
0.00	Constant
0 0.00 0.00	Array Constant

Function	Block Diagram Function (Array, Comparison and Cluster)
	Index array
	Max & Min
	Unbundle by name
	Bundle

Function	Block Diagram Function (Comparison)
>	Greater?
<	Less?
≥	Greater or equal?
≤	Less or equal?
?	Select
	Not

Block Diagram Terminal	Corresponding Front Panel Terminal	Control
TF	STOP	Boolean Control
TF	-	Write local variable of Boolean indicator or Boolean control
TF	-	Read local variable of Boolean indicator or Boolean control
TF	○	Boolean Indicator
DBL	0.00	Numeric control
DBL	-	Write local variable of numeric indicator or numeric control
DBL	-	Read local variable of numeric indicator or numeric control
DBL	0.00	Numeric Indicator
	0.00	Cluster
		File path

Structure	Block Diagram Function
	While loop
	Sequence
	Case

Chapter 1
Introduction

From the 1930s, process industries increasingly relied on automatic control. Techniques progressed from On-Off to Feedback in 1940s, [Bristol (1973)]. **Automatic process control** is concerned with maintaining process variables like temperature, pressure, flow, level, compositions etc at some desired operating value. Processes are dynamic in nature so changes always occur. If appropriate action is not taken in response then the important variables related to safety, product quality and production rates would not achieve design conditions. To achieve automatic process control, a control system must be designed and implemented. The three basic components of all control systems are Sensor/Transmitter, Controller, Final control element discussed briefly by [Pitt and Preece (1990), Smith and Corripio (1997)]. The types of control algorithms that are used in computerized control are discussed below.

1.1 Adaptive control:

An adaptive controller is a controller that can modify its behavior in response to changes in the dynamics of the process and the character of disturbances. The controller becomes nonlinear because of parameter adjustment mechanism. There have been a number of applications of adaptive feedback control since the mid-1950s. The early experiments were plagued by hardware problems. Systems implemented by using minicomputers appeared in the early 1970s. [Bristol (1973)] lists different types of adaptive systems like self-adaptive, optimizing or goal seeking, self-organizing, learning, pattern recognition. Adaptive techniques have been used in regular industrial controllers in early 1980s. Different design techniques for adaptive control with industrial applications of the same have been discussed by [Seborg et al. (1986)].

Numbers of applications of adaptive control have been discussed by [Astrom and Wittenmark (1995)]. The most widespread applications are in **Automatic tuning** of controllers. The parameters of controllers like PID controllers; which are majority of the regulators used in industry, are tuned automatically at the demand and after tuning the parameters are kept constant. The main advantage of using automatic tuner is that it simplifies tuning drastically and contributes to improved control quality. There are many ways of auto-tuning that have been proposed. The most common method is to make a simple experiment on the process.

The experiment can be done in open loop or closed loop. In open loop experiments, control rule is found from [Cohen-Coon (1953)]. The tuning experiments can also be done in closed loop. A typical example of this is the self-oscillating method of [Ziegler and Nichols (1942)]. It is difficult to automatize the experiment carried out by this method and perform it in such a way that the amplitude of the oscillation is kept under control; hence the parameters remain poorly tuned. A new autotuning method, relay feedback then proposed by [Astrom and Hagglund (1984)]. The relay feedback causes the process to oscillate with small and controlled amplitude. This autotuning method has been used in commercial autotuners since 1984. Many adaptive controllers are based on sampled controller with a sampling period magnitude same as that of the process time constant which introduce extra dead-time in the control loop because of the sampling. Such a controller is often not suitable for control loops subject to load disturbances, [McMillan (1986)]. Feedforward control is very useful but requires reasonably accurate process model. Relay feedback proven to be a convenient tool for rapid tuning of PID controllers, [Hagglund and Astrom (1991)]. Also Hagglund and Astrom discuss various examples like flow loop, temperature control, pulp density control and level control along with different design methods.

Another method for auto-tuning is to use an expert system to tune the controller. It is observed that actual implementation of control laws often incorporates a substantial amount of heuristic logic. Expert system provides a systematic approach for dealing with heuristic logic, [Astrom et al. (1986)]. The expert system waits for the set point changes or major load disturbances and then evaluates the performance of the closed-loop system. Properties such as damping, period of oscillation, and static gain are estimated. The controller parameters are changed according to built-in rules, [Astrom and Wittenmark (1995)].

The second application is of **Gain Scheduling**. It is an effective method for processes with predictable variations in dynamics. It has been found very useful in pH control, [Shinskey (1979)]. Gain scheduling is a nonlinear type feedback of special type; it has a linear controller whose parameters are changed as a function of operating conditions in preprogrammed way; it tabulates several sets of controller parameters, one for each operating point, [Hagglund and Astrom (1991)]. Autotuning greatly simplifies construction of gain scheduling. The hardware such as function generators and multipliers needed to implement gain scheduling was not available until recently and such components have been quite expensive to design and operate. However gain scheduling is easy to implement in computer-controlled systems, [Astrom and Wittenmark (1995)].

1.2 Fuzzy Logic Control:

In the forties the introduction of frequency characteristics and diagrams like Nyquist, Bode, Nichols to investigate stability of a system created an elegant and mathematically exact tool. Precise answers for gain and phase margin etc., to be chosen to achieve a good performance remained vague and subjective. Root locus method of Evans in the fifties suffered from the same ambiguity of exact values for the damping factors. Complex industrial plants such as batch chemical reaction processes often are difficult to control automatically. In such cases nonlinearity, time variance, stochastic disturbances have to be considered, but modeling methods for these become still complicated. Thus expert system, fuzzy logic controller, based on the heuristic logic came forward in an attempt to solve these problems, [Kickert and Lemke (1976)].

Fuzzy control is at present being extensively explored as a means to develop controllers for plants with complex and often not precisely known dynamics. The basis of a fuzzy control law is heuristic knowledge about a suitable control strategy. Fuzzy logic is a superset of conventional (Boolean) logic that has been extended to handle the concept of partial truth-truth-values between "completely true" and "completely false". Zadeh introduced it in the 1960's as a means to model the uncertainty of natural language. Zadeh says that rather than regarding fuzzy theory as a single theory, we should regard the process of "fuzzification"' as a methodology to generalize any specific theory from a crisp (discrete) to a continuous (fuzzy) form. This concept of fuzzy sets has been introduced by [Zadeh (1965)]. It is a mathematical way to represent vagueness in everyday life. Notions of inclusion, union, intersection, complement, relation, convexity etc. are extended to such sets and various properties of fuzzy sets established. The value zero is used to represent non-membership, and the value one is used to represent membership in fuzzy sets. The key elements in human thinking are not numbers but labels of fuzzy sets with fuzzy truths, fuzzy connectives and fuzzy rules of inference, discussed by [Zadeh (1973)]. [Didier et al. (1987)] suggest a relevant definition of mean value of a fuzzy number related to membership function.

The fuzzy control research was started by Mamdani's pioneering work, [Sugeno (1985)]. The starting point for fuzzy control has been the observation that human operators can solve many, often complex control tasks satisfactorily. Their approach is not based on any formal mathematical model & they do not use any precise algorithm. The strategy, which they use, can be described by qualitative statements as e.g.

"if the throughput is already low but temperature is still
too high and rising, then increase coolant flow quickly".

Here low, too high, rising, quickly do not identify a sharp interval of the respective variable but rather a range with some gray zone or "fuzziness", [Engell and Heckentheller (1995)].

Various theoretical models of fuzzy algorithm restricted to the class of discrete systems have been proposed and studied by [Zadeh (1968)]. Zadeh introduces a basic concept which though precise in nature, eventually proved to be of use in a wide variety of problems relating to information processing, control, pattern recognition, system identification, artificial intelligence and decision process involving incomplete or uncertain data. The earlier fuzzy controllers were rule based [Mamdani and Assilian (1975)]. [Braae and Rutherford (1979)] discuss algebraic model of the practical fuzzy logic controller and demonstrate its use in resolving engineering problems such as loop stability and beneficial controller adjustments, which are associated with practical applications of fuzzy logic controller. [Czogala and Pedrycz (1981)] formulate and discuss the problem of identification of fuzzy systems, identification and performance criteria, linear regression, and correlation analysis methods. Problems concerning complex processes are solved by fuzzy equation such as mutual identification and control, stabilization and control, control with constraints, generating linguistic control rules corresponding to fuzzy logic controller by [Czogala and Pedrycz (1982)]. [Takagi and Sugeno (1985)] present a mathematical tool to build a fuzzy model of the system. [Bezdek (1993)] provides notes on evolution of fuzzy models and fuzzy systems and also discusses examples. A self-learning relational fuzzy model-based controller is described by [Postlethwaite (1995)], which also sorts out advantages and drawbacks of the various existing types of fuzzy controller.

[Mamdani and Assilian (1975)] discuss the application of fuzzy control to steam engine. Fuzzy controllers have also been applied to a warm water plant, built in laboratory, [Kickert and Lemke (1976)]. [King and Mamdani (1977)] quotes that complex chemical processes such as blast furnaces, cement kiln and basic oxygen steelmaking are difficult to control automatically because of non-linearity, time varying behavior and poor quality of available measurements and discusses the example of boiler steam temperature control. In 1978 fuzzy logic controller was operating in closed loop on a rotary kiln in Denmark and it was the first successful test run on a full-scale industrial process. [Larsen (1980)] discusses this rotary kiln problem. Larson also gives an idea about combining fuzzy controllers for industrial processes with pure time delays with conventional analog or digital PID controllers. Activated sludge wastewater treatment process which is characterized by lack of relevant instrumentation, control goals that are not always clearly stated, the use of qualitative information in decision making and poorly understood basic biological behavior mechanisms make it an ideal candidate for fuzzy control as discussed by [Tong, Beck and Latten (1980)]. [Tong (1980)] reports on two attempts to construct fuzzy relational descriptions of experimental data using gas furnace data and river quality data as examples. [Graham et al. (1988)] describe a fuzzy adaptive controller, which uses fuzzy process model directly to obtain its control policy rather than by deriving explicit control rule set and it has been applied to a laboratory liquid level rig. [Engell and Heckentheler (1994)] discuss the level control of two-tank system. [Engell and Heckentheler (1995)] discuss the pH control example. [Remberg et al. (1995)] reports fuzzy control of distillation column.

1.3 Virtual Instrumentation:

1992 was the start of the PC based era in instrumentation. The expert opinion says the key to better measurements is software, [Grossman (1993)]. A modern instrument uses a general-purpose computer and a graphical software capability with various interfaces to communicate with and to control the hardware. This type of instrument uses the computer's extensive processing power to offer very high level of performance. Graphical programming language, which resulted from the concept of object oriented programming, uses data flow programming technique, relieves user of the responsibility of writing code, offers interesting and exciting programming possibilities. Its major advantage is simplicity because the user manipulates and wires icons in configuring his program.

Thus the graphical programming concept introduced the possibility of creating a new type of instrumentation, not in hardware but rather in software. This instrument was called **Virtual Instrument** (VI). This brought a change in instrumentation history whereby the user will no longer be dependent on the instrument vendor, who gave the user no option in modifying the instrument's functionality. With proper transducers, data-acquisition boards and other circuitry in place, an instrumentation system quickly comes together. The set up can be designed and redesigned from the computer's keyboard, [Brown (1992)].

The role of Virtual Instrumentation is to replace standard and custom laboratory instrumentation like voltmeters, frequency meters and counters; each device can be created on the PC as a VI with its own switches, knobs and display features. But it is not possible to replace mid range and high range instruments by VI and often not cost effective in low range instruments. Control strategies are easier and cheaper to implement in software than in hardware. Hence this software is also called as an Instrument.

1.3.1 Graphical programming for Virtual Instrumentation-LabVIEW:

The user interface of modern operating systems has become graphical. LabVIEW (Laboratory Virtual Instrument Engineering Workbench) is a program development environment. It is a professionally designed product with a great deal to offer the engineer embarking on a major project involving Data Acquisition, system design, simulation and eventual control, [Brown (1992)]. LabVIEW has been under development since 1986, is very mature, and has possibly the widest use in industry and academia of the true visual languages. Engineers are primarily goal oriented and a language is a tool to achieve the goal, which should be quickly and easily done, [Kauler (1993)]. Engineering should be holistic and integrative process and thus engineering education should be designed towards that end, [Ibrahim et al. (1993)]. Ibrahim et al.

also discusses various laboratory experiments using LabVIEW. LabVIEW has been also used in Physiological work as a measurement tool, by [Constable et al. (1991)].

The difference from the conventional type programming is that LabVIEW uses graphical programming language G, to create programs in block diagram form while others use text-based language. A VI has three main parts:

1. The **front panel** is the interactive user interface of a VI, so named because it simulates the front panel of a physical instrument.
2. The **block diagram** is VI's source code; constructed in LabVIEW's graphical programming language G.The components of a block diagram are lower level VIs, built in functions, constants and program execution control structures.
3. In order to use a VI as a subroutine in the block diagram on another VI, it must have an **icon** and a **connector**. A VI that is used within another VI is called a subVI and is analogous to a subroutine.

LabVIEW includes libraries for data acquisition, GPIB (General Purpose Instrument Bus) and serial port instrument control, data analysis, data presentation, and data storage. Thus LabVIEW, which can stimulate almost any type of measuring instrument in software, has made Virtual Instrumentation practical.

Other visual programming languages are HP-VEE, HI-VISUAL, Khoros, Serius and Prograph as reported by [Kauler (1993)].

1.4 Objective of the thesis:

After having a brief introduction of all the needs and techniques it's now appropriate to give an idea about the thesis that has been carried out. In this thesis using LabVIEW as a programming language an attempt is made to develop a Virtual Instrument for PC based process control. For this purpose a Level Control experiment setup is used. Using PID toolkit and Fuzzy Logic Design toolkit for G language different types of control algorithms namely Autotuning, Gain-Scheduling, Fuzzy Control are implemented and studied.

Chapter 2 discusses about the hardware required for measurements of process signals, computer interfacing and experimental setup. Chapter 3 explains about the control algorithms. Results and discussions are briefly discussed in chapter 4. Chapter 5 lists conclusions and recommendations for further work.

Chapter 2
Measurement, Interfacing and Experimental Set up

2.1 Sensors:

All Data Acquisition devices require a voltage signal. Therefore the basic need is to convert all signals to voltage signal. Sensor systems convert the physical quantity under measurement into electrical signal. Sensors usually consist of an actuating mechanism and a sensing element. Sensors can be classified into the categories such as self-generating analog, variable parameter analog frequency or pulse generating, digital, [S.Gupta (1989)]. Here in the experimental set up for liquid level variable parameter analog sensor is used which is a continuous rotation potentiometer. Thus it's a variable resistance-sensing element. The level transducer used here is of float type. This design uses the principle of a buoyant member, which floats on the surface of the liquid, and changes position as the liquid level varies. The float is connected to motion transducer which outputs electrical signal proportional to the liquid-level, [Doebelin (1975)]. Also the final control element, motorized valve, has a potentiometer which gives output voltage corresponding to valve position.

2.2 Data Acquisition (DAQ):

Many real world sensors and transducers output signals that must be conditioned before a DAQ device can effectively and accurately acquire the signal. This front-end preprocessing, which is generally referred to as **signal conditioning,** includes functions such as signal amplification, filtering, electrical isolation, and multiplexing, [NI-AN048 (1997)]. In order to use the digital computer for any application other than data processing, it has to be connected to some instruments, sensors, and actuators. This connection is called the **interface.** Analog Signals come in continuous form. Thus it is necessary to have some device that can translate these analog signals to digital form to be intelligible to the computer and vice versa. The interfacing may be unidirectional or birectional, [S.Gupta (1989)]. Data logging requires a unidirectional connection since the computer is simply a "listener" and only takes in data from sensors. On the other hand control requires a bi-directional connection since the computer is a "talker (AO)" as well as "listener (AI)". The computer has to acquire data from sensors (listener) and then has to take corrective action (talker) according to control algorithm sitting in the software.

2.3 DAQ Hardware and Configuration:

In the present experimental set up Signal Conditioning eXtension for Instrumentation (SCXI) product line by National Instruments is used as Data Acquisition hardware. SCXI is a signal conditioning and instrumentation front-end DAQ device. An SCXI system consists of an SCXI chassis that houses one or more signal conditioning modules that multiplex, amplify, isolate, and condition both analog and digital signals. The SCXI system then passes the conditioned signals to a single DAQ device for acquisition directly into the PC, as shown in Figure 2.1.

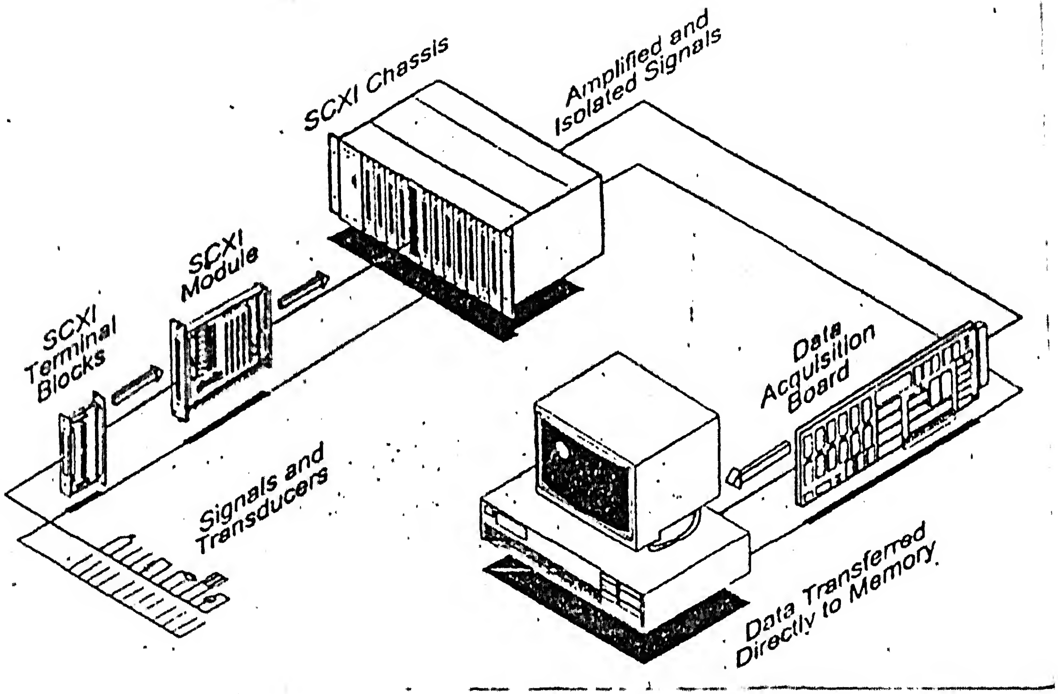


Figure 2.1: A PC based DAQ System with SCXI

(Reproduced from [NI-AN048, (1997)])

2.3.1 Configuration of SCXI modules:

Installation and configuration procedure of Chassis and Modules are discussed in detail in User Manual. Also see appendix A, which gives steps for installation. For present study the configured values were as follows,

1. SCXI-1001 (SCXI Chassis):

Chassis ID: 1

Chassis Address: 0

Module 1: SCXI-1200, 12-bit acquisition and control

Module 2: SCXI-1100, 32-channel multiplexer amplifier

Module 3: SCXI-1124, 6-channel D/A converter

Module 4: SCXI-1124, 6-channel D/A converter

Modules 5 to 12: empty

2. SCXI-1200:

Device No.: 1

Resources (Connected to): Parallel Port-LPT1

AI Polarity/Range: 0 to +10 V

AI Mode: Nonreferenced Single ended

AO Polarity: Unipolar

Accessory (Terminal Block): SCXI-1302 (Not Used)

3. SCXI-1100:

Connected to: SCXI-1200

Operating mode: Multiplexed

Gain: 1000

Filter: None

Accessory (Terminal Block): SCXI-1300 (AI connections)

4. SCXI-1124:

Connected to: SCXI-1200

Operating Mode: Multiplexed

Accessory (Terminal Block): SCXI-1325 (AO connections)

The SCXI-1200 DAQ module is used as interfacing, acquisition and control card. It is connected to parallel port of PC, LPT1. The signal-conditioning ADC (Analog to Digital) module used is SCXI-1100 called as a multiplexer amplifier. The DAC (Digital to Analog) module is SCXI-1124. All these modules are housed in SCXI-1001 12-slot chassis. Both the devices, SCXI-1100 and SCXI-1124 communicate to the PC through SCXI-1200.

2.4 Experimental set up:

The experimental set up used is a one tank SISO system. Schematic sketch is shown in Figure 2.2. Water is pumped from tank T2 to tank T1 through rotameter F1. Between the rotameter F1 and tank T1 motorized valve is mounted. Valve V1 serves as manual adjustment of water flow to T1. V2 serves as bypass valve. V3 is for draining the water from T2. V4 is recycle valve to tank T2. The flow from V4 depends upon the water level in T1. The aim here is to control the water level in T1 at some desired value.

A level sensor mounted on the T1 gives electrical signal proportional to the level in the T1. The level signal is fed to the interfacing device (SCXI System) and then the conditioned signal is fed to the computer. Also the position of the valve (stem position) is sensed through a potentiometer and then fed to computer. The control algorithm present in the computer processes these signals and gives an output to the motorized valve, again through SCXI System. The inlet flow rate acts as a disturbance to the process.

Motorized Valve is the final control element, which implements the decision taken by the control. To do this assignment a motor control algorithm is developed, the basic idea of which follows from the following information of the valve. It contains a stem on which a diaphragm is mounted. A DC motor rotates the stem. The stem moves up to close the valve; the motor rotates in clockwise direction and vice-versa. A potentiometer connected to the stem of the valve gives voltage corresponding to the stem position. The motorized valve is having a 5-pin connection, among which two pins are for power supply to the DC motor, third one acts as a common ground for the fourth and fifth pin. The fourth pin is for power supply to the potentiometer and fifth one is for reading potentiometer voltage.

2.4.1 Acquisition and Analysis of process signals:

In the present system analog inputs (AI) are two in number. First one is voltage signal from level sensor used for level measurement. Second one is voltage signal from potentiometer of motorized valve, which gives the position of stem of the valve. Both the signals are connected to SCXI-1100 module through SCXI-1300 terminal block. The relationship between voltage signal and corresponding liquid level is found to be linear. Also the relationship between voltage signal and corresponding valve position is linear. See Appendix B for calibration curve. The data coming in the form of voltage signals is analyzed and converted in appropriate engineering units, see Figure 2.4, 2.5. For level the voltage signal is converted into mm as well as percentage of the entire range; while for motorized valve the voltage signal is converted to percentage of the entire range. To convert level into mm spline interpolation method is used. Figure 2.3 shows the flowchart. Figure 2.4 and 2.5 show the conversion of level sensor data from volts to and vice versa respectively using spline interpolation method.

The VI: *Read from file* contains the calibrated data of level sensor, (see appendix A). User can refresh this data based on latest calibration. One can check the level sensor voltage using VI: *voltage measurement* located in the *Search Examples/Data Acquisition/Transducer Measurements* in LabVIEW help menu. The same applies for motorized valve. The user can also refresh maximum and minimum values of both the voltage signals. The subVI: *PV filter*, shown in Figure 2.7, which can be modified by putting different filtering parameters. Appendix C lists all the associated subVIs used.

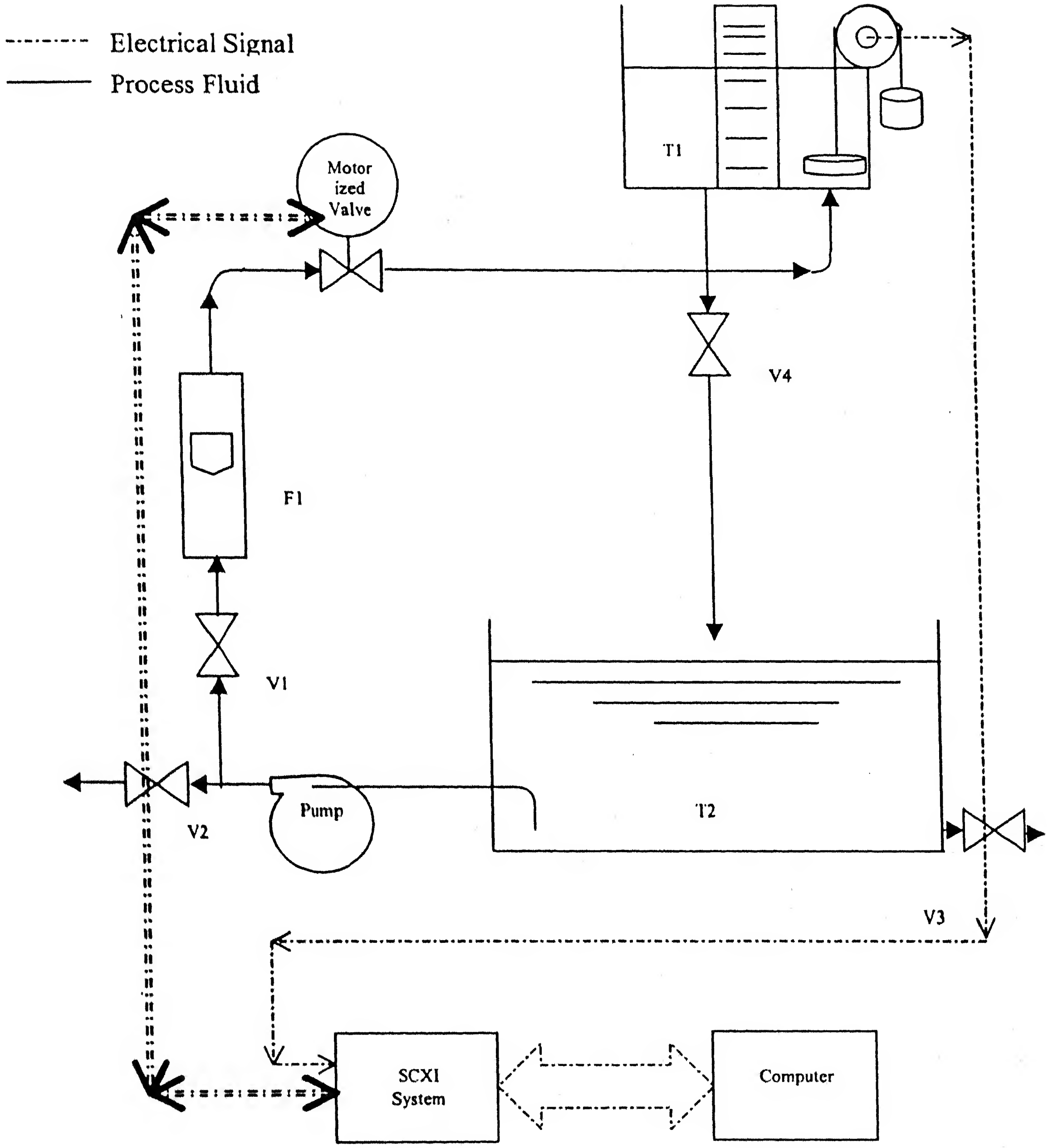


Figure 2.2: Schematic diagram of experimental set up.

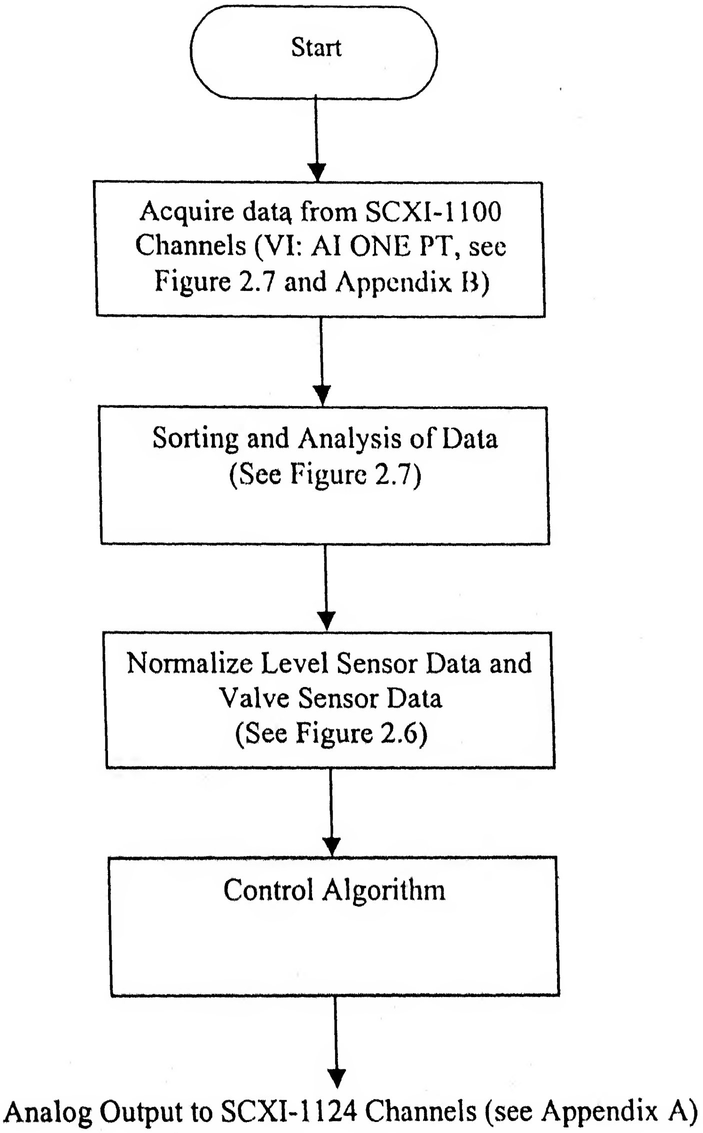
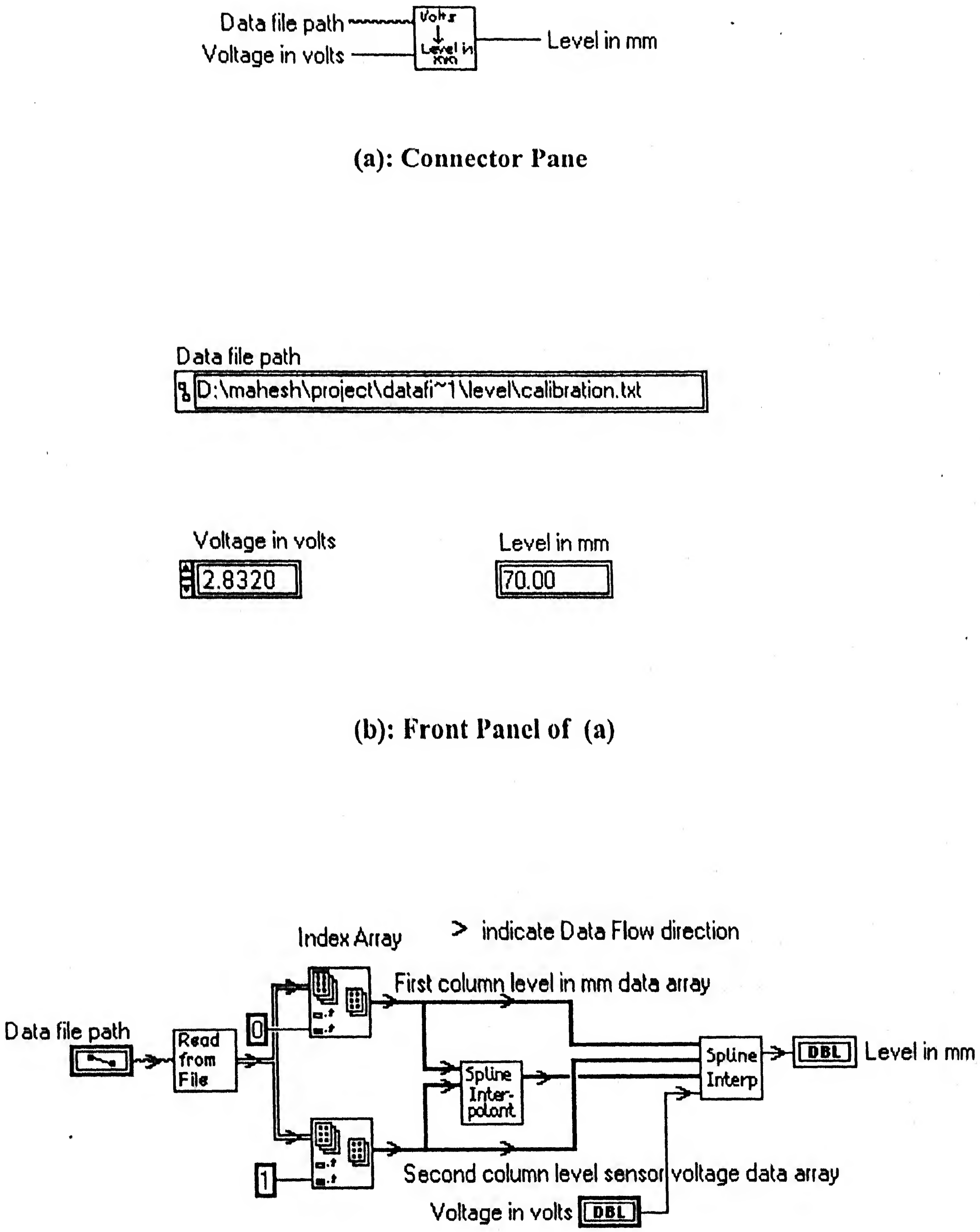

Start
Acquire data from SCXI-1100 Channels (VI: AI ONE PT, see Figure 2.7 and Appendix B)
Sorting and Analysis of Data (See Figure 2.7)
Normalize Level Sensor Data and Valve Sensor Data (See Figure 2.6)
Control Algorithm
Analog Output to SCXI-1124 Channels (see Appendix A)

Figure 2.3: Data analysis flowchart.



(a): Connector Pane

Data file path: D:\mahesh\project\datafi~1\level\calibration.txt

Voltage in volts: 2.8320

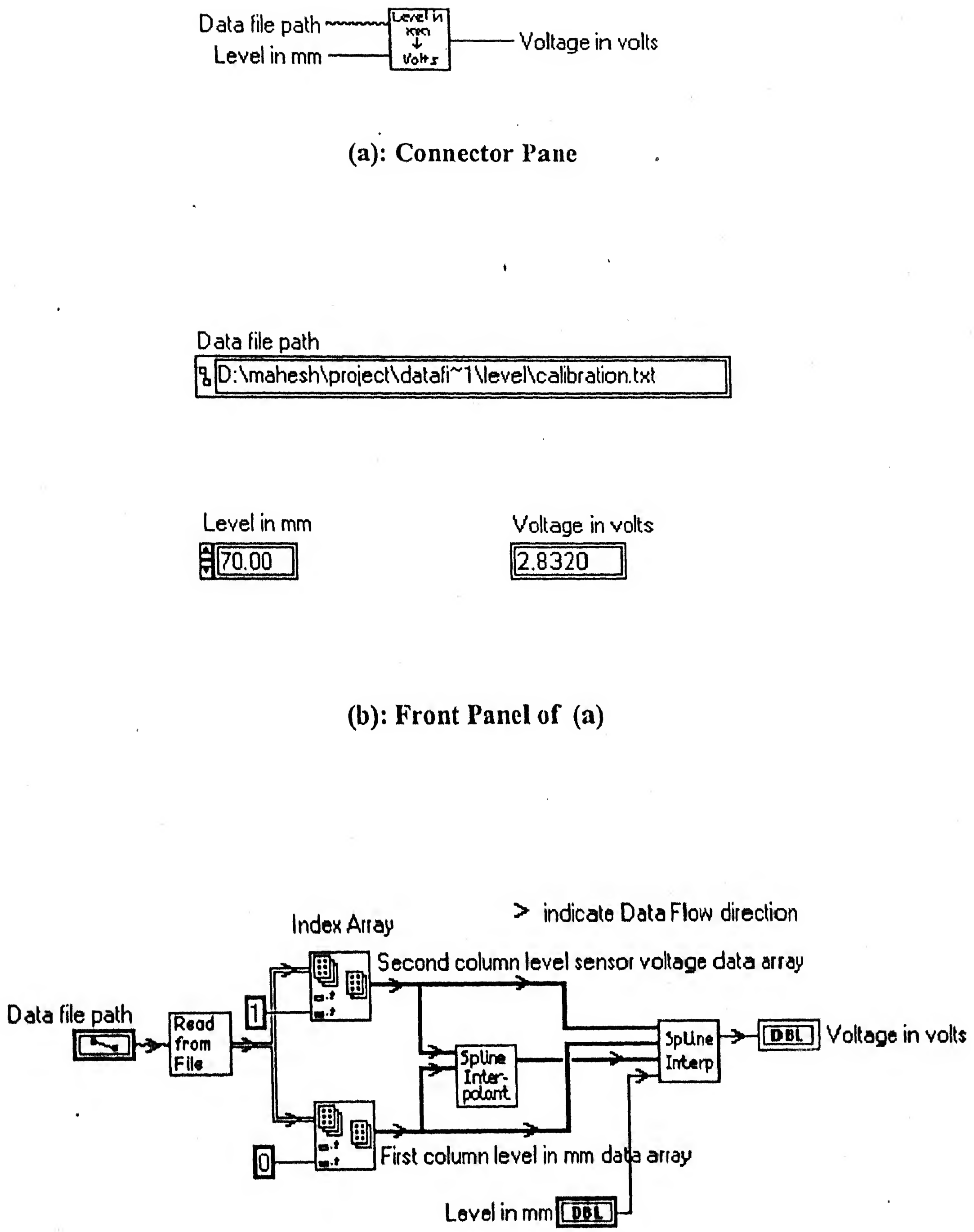
Level in mm: 70.00

(b): Front Panel of (a)

Index Array
> indicate Data Flow direction
First column level in mm data array
Data file path
Read from File
0
1
Spline Inter-polant
Spline Interp
DBL
Level in mm
Second column level sensor voltage data array
Voltage in volts
DBL

(c): Block Diagram of (a)

Figure 2.4: VI for conversion of level sensor data from volts to mm: *v to l.vi*



(a): Connector Pane

Data file path
D:\mahesh\project\datafi~1\level\calibration.txt

Level in mm
70.00

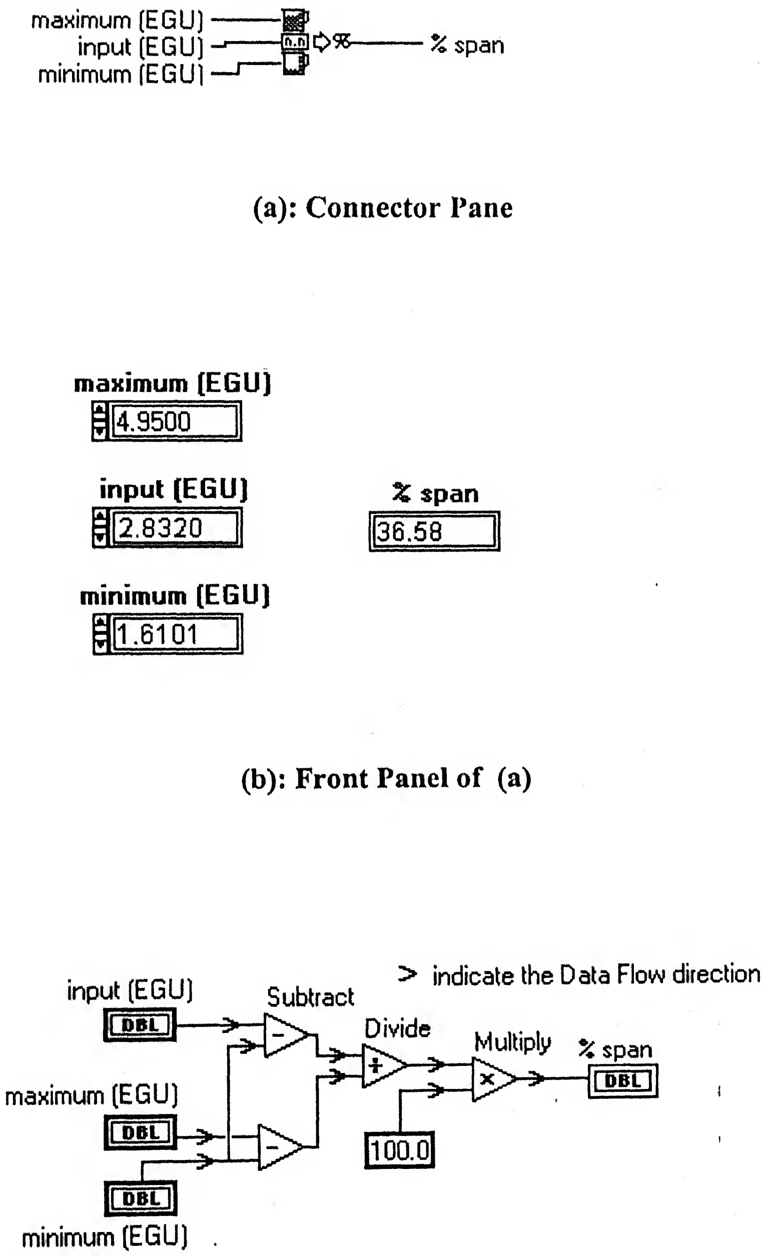
Voltage in volts
2.8320

(b): Front Panel of (a)

Index Array
> indicate Data Flow direction
Second column level sensor voltage data array
Data file path
Read from File
1
0
Spline Inter-polant
Spline Interp
DBL
Voltage in volts
First column level in mm data array
Level in mm
DBL

(c): Block Diagram of (a)

Figure 2.5: VI for conversion of setpoint data from mm to volts: *l to v.vi*



(a): Connector Pane

(b): Front Panel of (a)

(c): Block Diagram of (a)

Figure 2.6: VI for normalization of voltage data: ***EGU to %.vi***

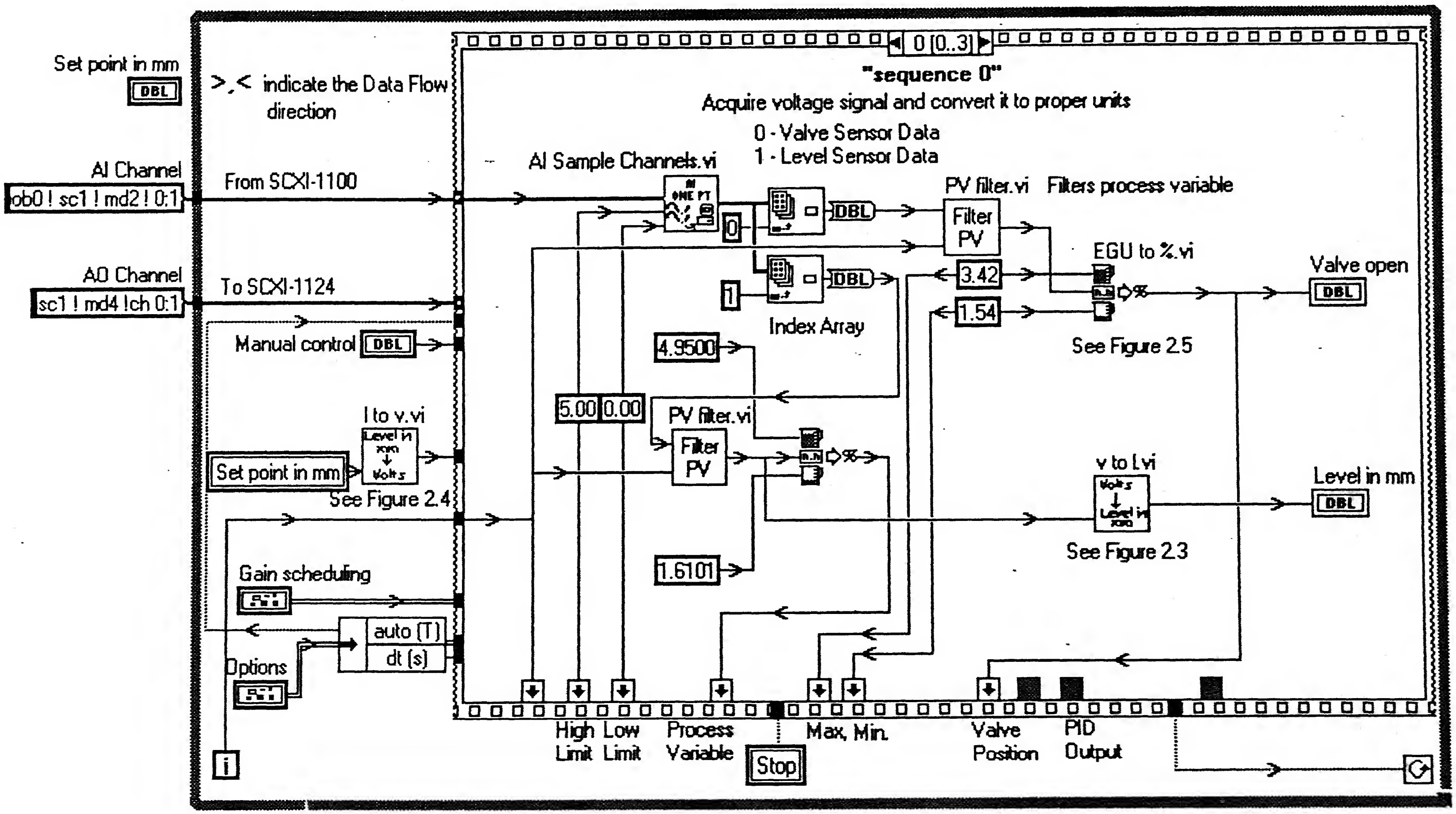


Figure 2.7: Block Diagram for Acquisition and Conversion of electrical signals

Chapter 3
Control Algorithms

3.1 The Autotuning PID Algorithm:

The PID VIs uses positional PID algorithm, see Appendix E for discretized equations used. [Shinskey (1979)] describes continuous nonlinear model by expression,

$$u(t) = k_c f|e|\left(e(t) + \frac{1}{\tau_I}\int edt - \tau_D \frac{dPV_f}{dt}\right) \tag{3.1}$$

where,

$$f|e| = L + \frac{(1-L)|e|}{100} \tag{3.2}$$

where L is an adjustable parameter representing *linearity*, which ranges from 0 to 1 ($L =$ 1 linear) and e is expressed in percent.

And error e is expressed for proportional control as, [Astrom and Wittenmark (1995)],

$$e(t) = \beta \cdot SP - PV_f(t) \tag{3.3}$$

where β is a *setpoint factor* either 0 or 1 tracking either *load changes* or *setpoint changes*.

PV_f refers to the filtered process variable. *Filtering* is required to eliminate noise from the measured signal. A linear *digital filter* can expressed as, [Smith and Corripio (1997)],

$$PV_f = aPV + (1-a)PV \tag{3.4}$$

where a is called the filtering parameter, with a values $0 \le a \le 1$.

3.1.1 The Autotuning Algorithm:

Autotuning relay feedback method is used here. The autotuner principle is shown in figure 3.1. When operator decides to tune the controller, he is required to simply press a button. This switches out PID algorithm and replaces it with a nonlinear function, which can be described as a relay with hysteresis. The function used here is setpoint relay. In this method when autotuning is started the setpoint is changed corresponding to relay amplitude. This causes process to oscillate with controlled amplitude. The *frequency of the limit cycle* is approximately the ultimate frequency where the process has a phase lag of 180^0. The *ratio* of the *amplitude of the limit cycle* and the *relay* *amplitude* is approximately the *process gain* at that frequency. The parameters of PID controller then determined by Ziegler and Nichols closed loop method, see Appendix E.

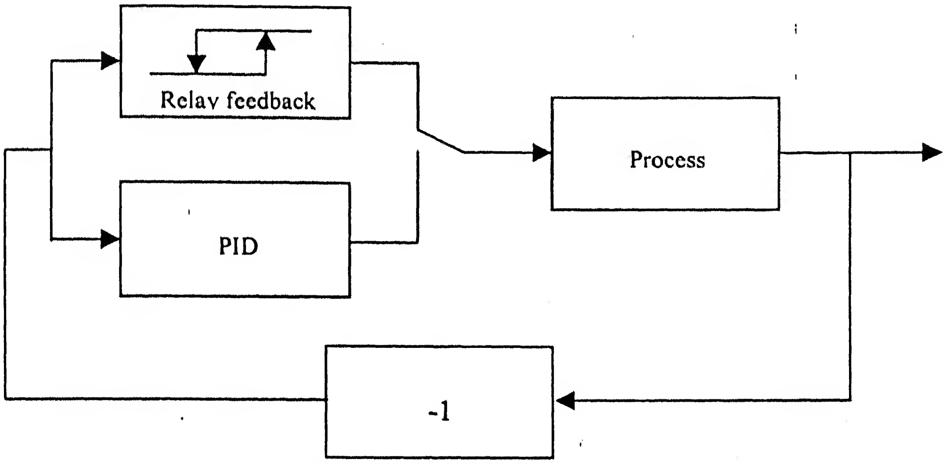


Figure 3.1: The autotuning principle

3.1.2 Gain Scheduling:

A gain scheduling is a table with several sets of controller parameters, one for each operating point. A reference signal, which is related to the nonlinearity, determines when to switch from one set of controller parameters to another. If the nonlinearity is caused by a nonlinear valve, the control signal should be used to select controller parameters. If the nonlinearity is caused by a nonlinear sensor, the measurement signal should be used to select controller parameters. If process dynamics are constant then a controller with constant parameters is chosen. If process dynamics are varying, then the controller with changing parameters is used which can be achieved by introducing gain scheduling.

3.2 Motorized Valve Control:

A motorized valve finally implements the controller decisions calculated from above discussed algorithms. The motorized valve consists of a constant speed reversible motor, which is used to drive a valve. This type of operator has three states, drive upward (closing), stop and drive downward (closing). Thus the position of valve is controlled as follows. First the potentiometer signal is measured and normalized. After that it is compared with controller output. If the controller output is less than current valve position (percent valve open), the power supply to the motor is adjusted so that it moves in a clockwise direction so as to close the valve. It is then allowed to move until it equals the controller output and then it is stopped. The similar procedure but in the

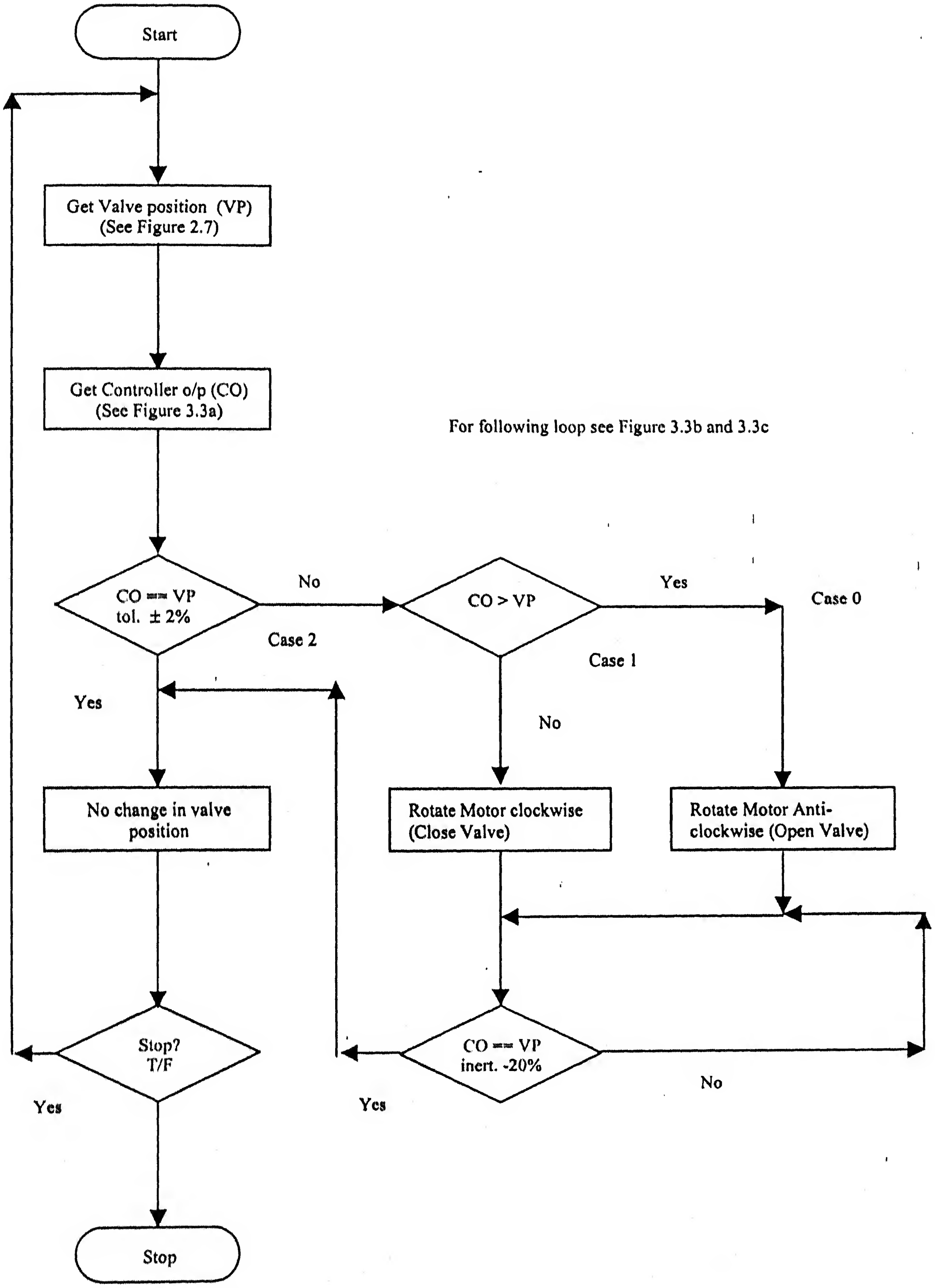


Figure 3.2: Motorized Valve Control flowchart.

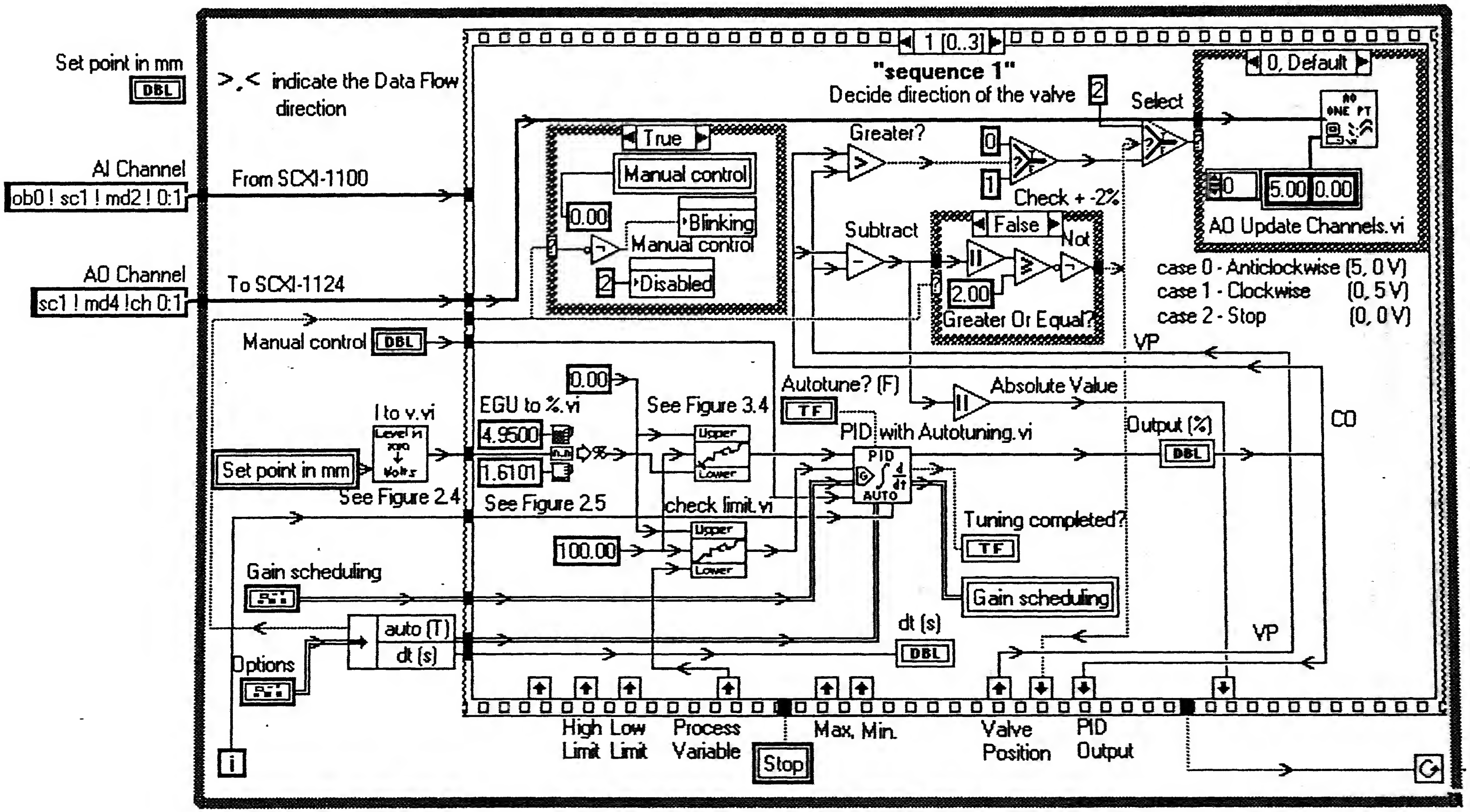


Figure 3.3(a): Block diagram to decide movement of the valve.

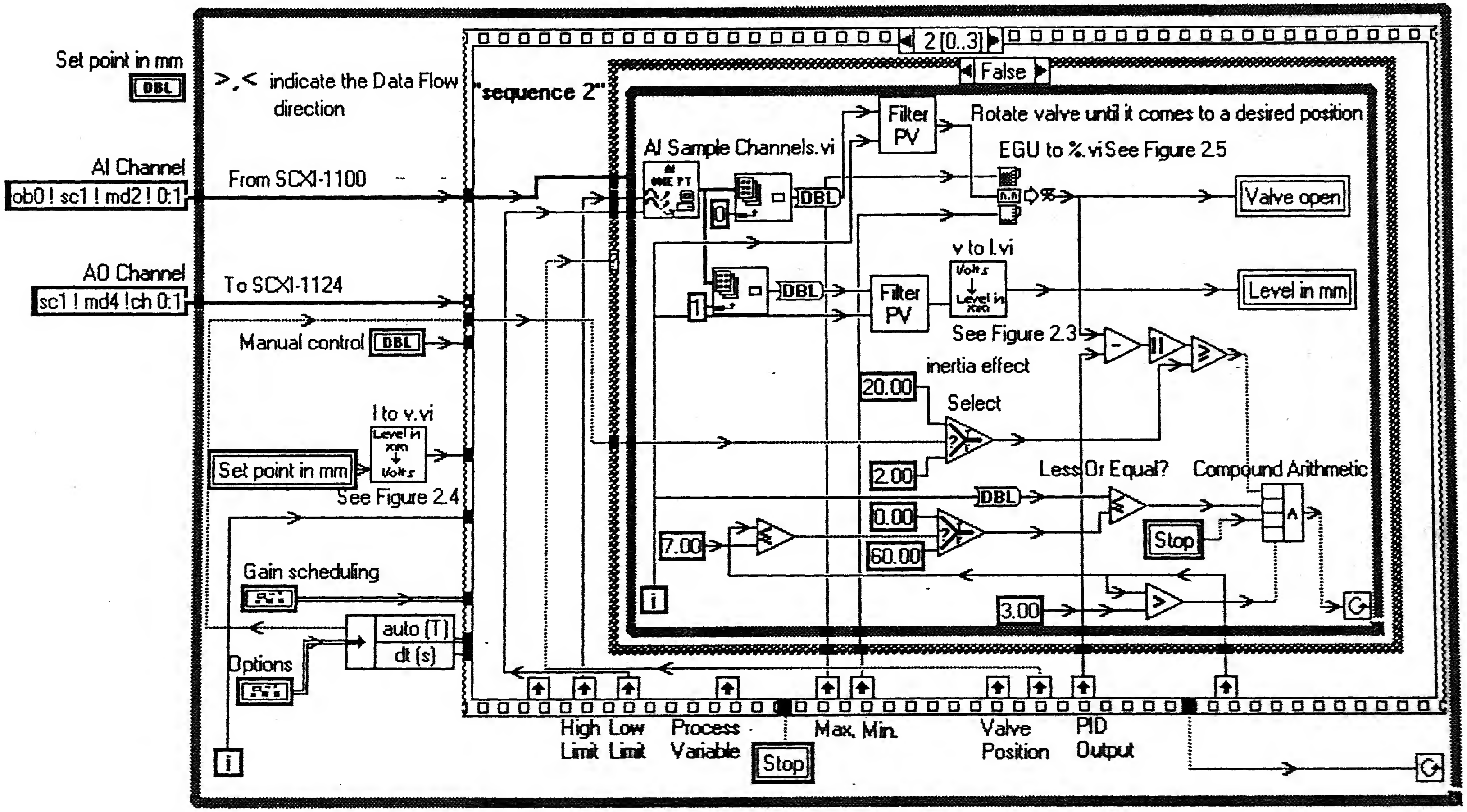


Figure 3.3(b): Block Diagram to decide the extent of movement of valve.

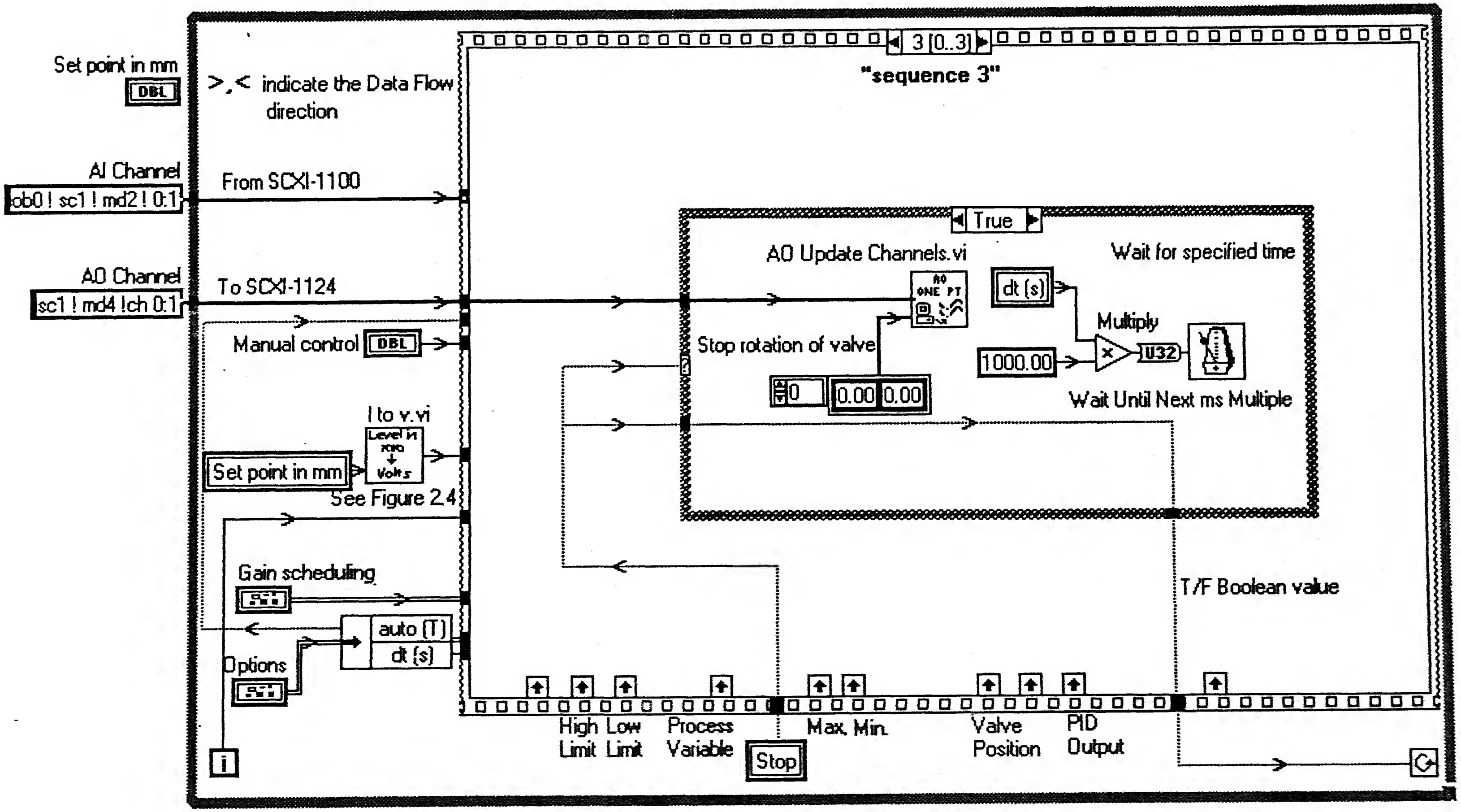
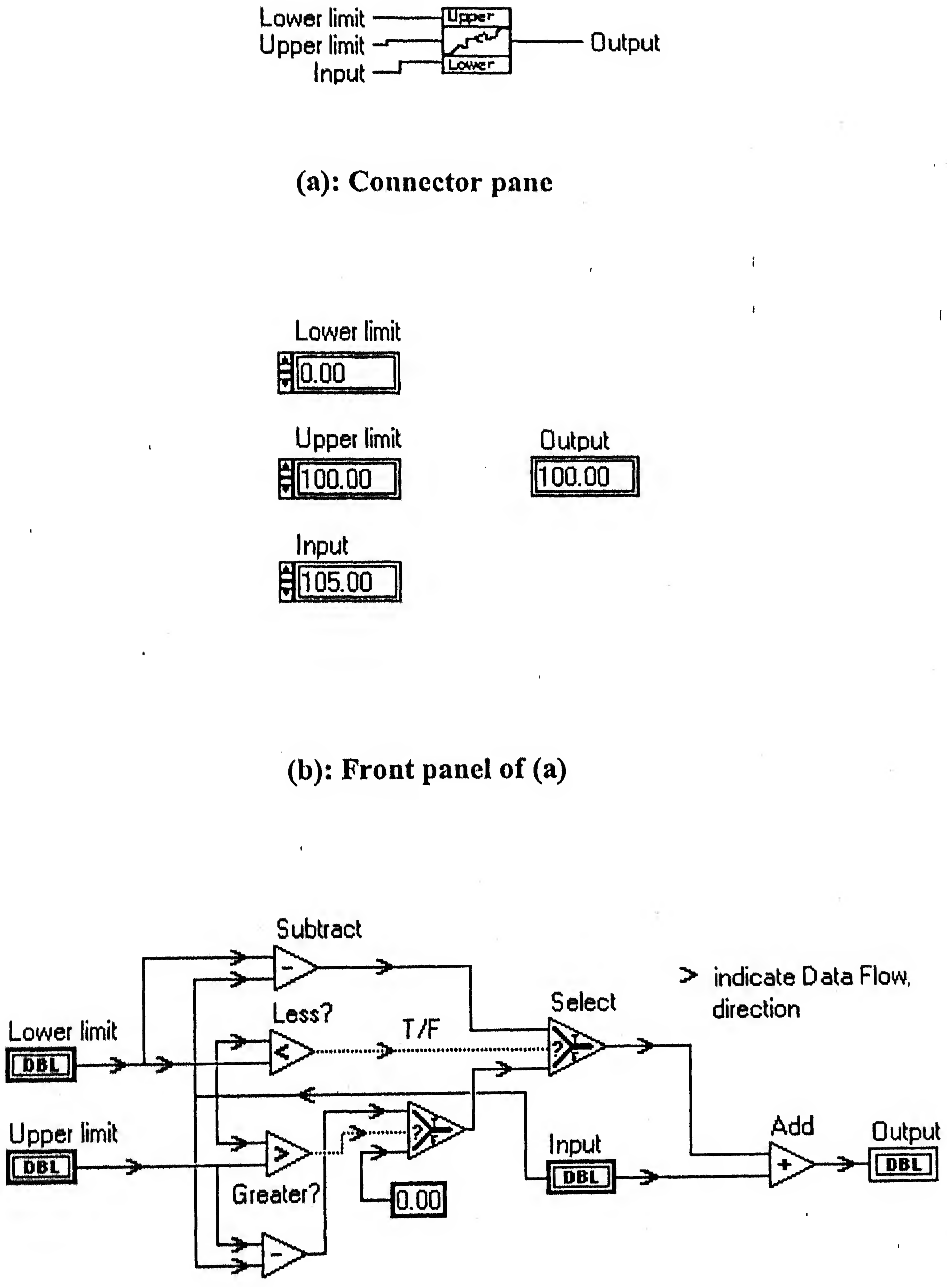


Figure 3.3(c): Block Diagram to stop the valve till specified time.



(a): Connector pane

(b): Front panel of (a)

(c): Block diagram of (a)

Figure 3.4: VI for checking upper and lower limits: *check limit.vi*

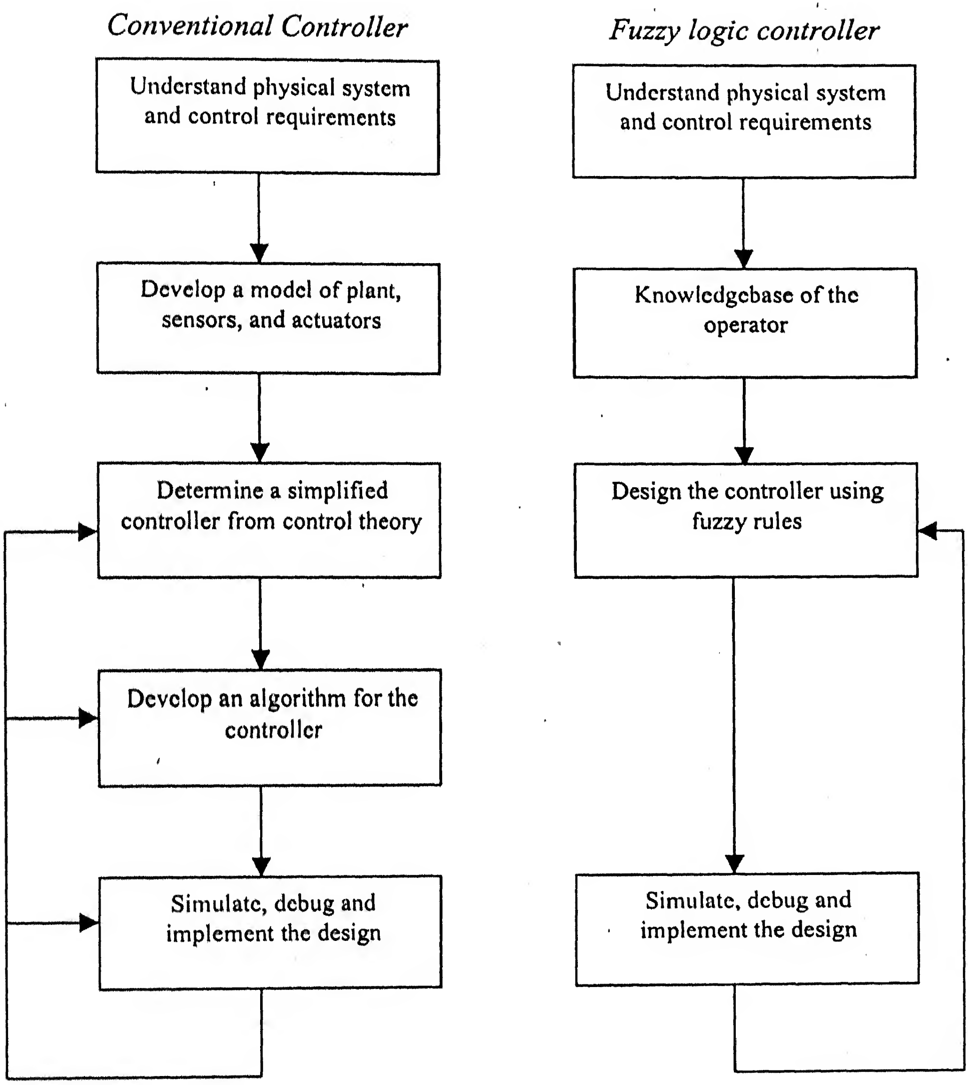
direction is applied to other case where controller output is greater than current valve position. The valve's inertia effect is taken into account by stopping the valve at lower value than exact. The algorithm of motor control is given in Figure 3.2. The Figures 3.3a, 3.3b and 3.3c show the Block Diagram of the Autotuning PID Level Control, which incorporates former discussion, see Appendix D. Figure 3.4 shows block diagram of subVI *check limit.vi*, which restricts value to a specified limit.

3.3 Fuzzy Logic Control:

The fuzzy logic controller is studied for present case of controlling liquid level. The figure 3.5 shows difference in the evaluation of fuzzy logic controller from conventional PID type controller. It can be seen that modeling work, which is a complicated one, is not required. Simply based on fuzzy statements like *if-then*, database of fuzzy rules is prepared. In fuzzy control there are three basic steps involved. The first is *fuzzification* converting *physical values* into *heuristics language* called as *antecedence*. Then the second step involved is drawing *inference* out of *antecedence*. And third is converting the *inference* based on *heuristic language* into *real physical values* called as *defuzzification* process, [Engell and Heckentheller (1995)]. The Figure 3.6 shows the flow of information.

Step1:

If dh/dt is *big* and *e* is *small* then increase the flow *strongly*. This is a fuzzy statement made by human thought. In fuzzy logic going from binary 0/1 *membership function* to a continuous function solves this type of statement. Thus the degree of truth to which a measurement value of a technical quantity satisfies the linguistic concept of a certain term of a linguistic variable is called degree of membership (in the range of 0 to 1). There are infinitely many possible choices of membership functions μ, which denote the degree to which a continuous variable has the property of interest. For a continuous variable degree of membership can be modeled by a mathematical function, [Zimmerman (1987)]. The normalized standard membership functions are of Z-type, $\wedge$-type (triangular shape), Π-type (trapezoidal shape), and S-type membership function shapes. These are shown in following Figure 3.7. This is how the *fuzzification* is done.



Conventional Controller
Understand physical system and control requirements
Develop a model of plant, sensors, and actuators
Determine a simplified controller from control theory
Develop an algorithm for the controller
Simulate, debug and implement the design
Fuzzy logic controller
Understand physical system and control requirements
Knowledgebase of the operator
Design the controller using fuzzy rules
Simulate, debug and implement the design

Figure 3.5: Conventional and Fuzzy design methodology

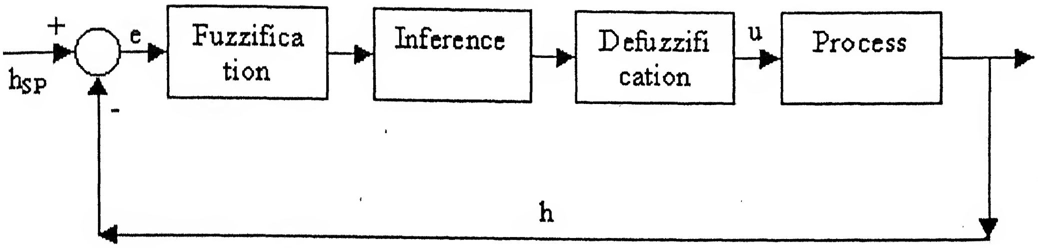


Figure 3.6: Fuzzy Controller

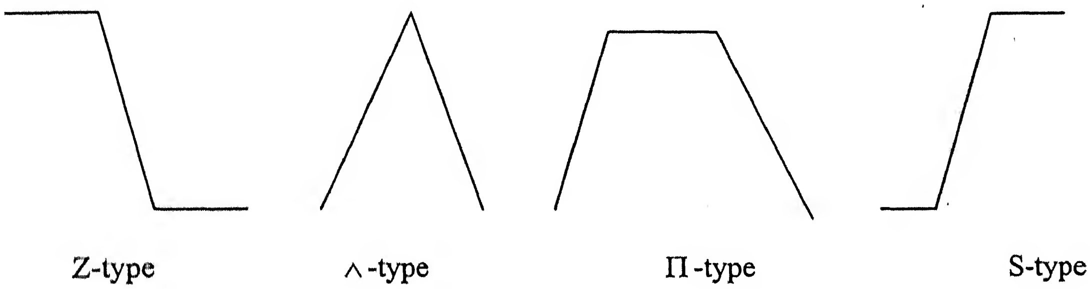


Figure 3.7: Shapes of Standard Membership Functions

Step2:

Inference is an indirect step, which springs from fuzzification. In the inference step the relation between input variables and output variables is established. The next ingredient is to formulate and evaluate logical statements.

Step3:

The most common method for *defuzzification* is *max-min-inference* and computation of *center of mass* also called as *center of gravity*.

Membership function for the output *u* is calculated by,

$$\mu(u) = \max\min\left\{\mu(R1), \mu_{concl.}(u)\right\} \tag{3.5}$$

$$u_{out} = \frac{\int \mu(u)\cdot u du}{\int \mu(u) du} \tag{3.6}$$

This involves lot of computation. Therefore a simplified method is used: the qualitative conclusions are identified with real values, singletons, which can be taken as the centers of mass of the individual fuzzy sets, rather than fuzzy sets and the output is the weighted average of these real values:

Rules i to n,

$$u_{out} = \frac{\sum_{i=1}^{n} \mu(u_i)\cdot u_i}{\sum_{i=1}^{n} \mu(u_i)} \tag{3.7}$$

3.3.1 Mathematical Approach:

In this section mathematical approach to the fuzzy identification of systems and fuzzy modeling is presented, [Takagi T. and Sugeno M. (1985)].

1. Format of fuzzy implication and reasoning algorithm:

Membership function of a fuzzy set A is denoted as A (x), x $\in$ X.All the fuzzy sets are associated with linear membership functions. Thus a membership function is characterized by two parameters giving the greatest grade 1 and the least grade 0.The truth value of a proposition x_1 *is A and* x_2 *is B* is expressed by,

$$\left| \ x_1 \text{ is } A \text{ and } x_2 \text{ is } B \ \right| = A(x_1) \wedge B(x_2) \tag{3.8}$$

2. Format of Implication or Rule:

A fuzzy implication R is of the format

$$\text{R: If } f\left(x_1 \text{ is } A_1, \ldots, x_k \text{ is } A_k\right) \text{ then } u = g(x_1, \ldots x_k) \tag{3.9}$$

where u is variable of the consequence whose value is inferred.

$x_1, \cdots, x_k$ are variables of the linguistic that appear both in the rules and implicitly as part of consequence.

$A_1, \cdots, A_k$ are fuzzy sets with linear membership functions representing a fuzzy subspace in which the implication or rule R can be applied for reasoning.

f is a logical function that connects the propositions in the linguistic statements.

g is function that implies the value of u when $x_1, \cdots, x_k$ satisfies the linguistic statements.

3. Algorithm of identification:

Consider a fuzzy model consisting of some number of implications or rules that are of the format

If x_1 is A_1 and ... and x_k is A_k

Then $u = p_0 + p_1 \cdot x_1 + \cdots + p_k \cdot x_k$

Characterized by connective *and* and a *linear equation*. For identification following three parameters are to be determined using the input output data of an objective system. These are namely $(x_1, \cdots, x_k), (A_1, \cdots, A_k), (p_0, \cdots, p_k)$.

a) Choice of Linguistic Variables:

First a combination of linguistic variables is chosen out of possible input variables. Next the optimum linguistic and consequence variables are identified according to following steps.

i) The errors between the output values of the model and the output data of the objective system are calculated.

ii) Then improve the choice of linguistic variables so that performance index is decreased, which is defined as the root mean square of the output errors.

b) Linguistic parameter identification:

In this step the optimum values of linguistic parameters are searched for the linguistic variables chosen at step. Assuming the values of linguistic variables $x_1, \cdots, x_k$ the optimum consequence parameters $p_0, p_1, \cdots, p_k$ together with the performance index can be obtained. This leads to a minimized performance index or summation of the least squares error.

c) Consequence parameters identification:

The consequence parameters that give the least performance index are searched by the least squares method for the given linguistic variables in step 1 and parameters in step 2.

Let a system be represented by the following implications,

R^1 If x_1 is $A_1^1, \cdots$, and x_k is A_k^1

then $u_j = p_0^1 + p_1^1 \cdot x_{1j} + \cdots + p_k^1 \cdot x_{kj}$

$\vdots \qquad \vdots$

R^n If x_1 is $A_1^n, \cdots$, and x_k is A_k^n

then $u_j = p_0^n + p_1^n \cdot x_{1j} + \cdots + p_k^n \cdot x_{kj}$

then the output u for $(x_1, \cdots, x_k)$ is obtained as,

$$u_j = \frac{\sum_{l=1}^{n} \left(A_{1j}^l \left(x_{1j} \right) \wedge \cdots \wedge A_{kj}^l \left(x_{kj} \right) \right) \cdot \left(p_0^l + p_1^l \cdot x_{1j} + \cdots + p_k^l \cdot x_{kj} \right)}{\sum_{l=1}^{n} \left(A_{i1}^l \left(x_{1j} \right) \wedge \cdots \wedge A_{kj}^l \left(x_{kj} \right) \right)} \qquad j = 1, \cdots, m \qquad (3.10)$$

n is number of rules.

m is number of input output data sets.

where $\wedge$ denotes the picking of minimum value.

Let β_{ij} be the truth value for j^{th} data set and i^{th} rule then,

$$\beta_{ij} = \frac{A_{i1}^l \left(x_{1j} \right) \wedge \cdots \wedge A_{ik}^l \left(x_{kj} \right)}{\sum_{l=1}^{n} A_{1j}^l \left(x_{ij} \right) \wedge \cdots \wedge A_{kj}^l \left(x_{kj} \right)} \qquad (3.11)$$

then for the m data sets,

$$u_j = \sum_{l}^{n} \beta_{ij} \left(p_0^l + p_1^l \cdot x_{1j} + p_2^l \cdot x_{2j} + \cdots + p_k^l \cdot x_{kj} \right) \qquad j = 1, \cdots, m \qquad (3.12)$$

which can be expressed in matrix form using,

$$X = \Bigg[\begin{bmatrix} \beta_{11} & \beta_{21} & \cdots & \beta_{n1} \\ \beta_{12} & \beta_{22} & \cdots & \beta_{n2} \\ \vdots & \vdots & \vdots & \vdots \\ \beta_{1m} & \beta_{2m} & \cdots & \beta_{nm} \end{bmatrix} \begin{bmatrix} x_{11} \cdot \beta_{11} & x_{11} \cdot \beta_{21} & \cdots & x_{11} \cdot \beta_{n1} \\ x_{12} \cdot \beta_{12} & x_{12} \cdot \beta_{22} & \cdots & x_{12} \cdot \beta_{n2} \\ \vdots & \vdots & \vdots & \vdots \\ x_{1m} \cdot \beta_{1m} & x_{1m} \cdot \beta_{2m} & \cdots & x_{1m} \cdot \beta_{nm} \end{bmatrix} \cdots \begin{bmatrix} x_{k1} \cdot \beta_{11} & \cdots & x_{k1} \cdot \beta_{n1} \\ x_{k2} \cdot \beta_{12} & \cdots & x_{k2} \cdot \beta_{n2} \\ \vdots & \vdots & \vdots \\ x_{km} \cdot \beta_{1m} & \cdots & x_{km} \cdot \beta_{nm} \end{bmatrix} \Bigg]_{m \times (1+k)n}$$

(3.13)

$$\text{and,} \quad U = \begin{bmatrix} u_1 & u_2 & \cdots & u_m \end{bmatrix}_{m \times 1}^T \qquad (3.14)$$

$$\text{then,} \qquad U = XP \qquad (3.15)$$

where P is the parameter vector given by,

$$P = \begin{bmatrix} p_0^1 & \cdots & p_0^n & ; & p_1^1 & \cdots & p_1^n & ; & p_k^1 & \cdots & p_k^n \end{bmatrix}_{(1+k)n}^T \qquad (3.16)$$

The parameter vector p is calculated by least square minimization,

$$P = \left(X^T X \right)^{-1} X^T U \qquad (3.17)$$

3.4 Fuzzy knowledge based controller:

Fuzzy knowledge based controller (FKBC) is a one type of direct expert control system (DECS). The idea behind a FKBC is to build *knowledge based system* (KBC), which employs the *if-then* production rules. The knowledge base of a FKBC consists of *database* and *rule base*. The basic function of the *database* is to provide the necessary information for the proper functioning of the fuzzification. The basic function of the *rule base* is to present represent in a structured way the control policy of an *experienced process operator/* or *control engineer* in the form of set of *production of rules*.

3.4.1 Rule Base:

The design parameters of the rule base include,

1. Choice of Variables and Content of Rules:

The choice of designing a P (Propotional), PI (Propotional-Integral) like FKBC implies the choice of process state and control output variables, as well as the content of the rule-antecedent (*if*-part) and the rule- consequent (*then*-part) for each of the rules. The process state variables representing the contents of the antecedent are selected amongst error e *and* change of error Δe(or rate of change of error). The control output variables representing the rule-consequent are selected amongst control output u or change of control output Δu .

P-like FKBC:

A conventional P-controller uses an analytical expression of the following form to compute the control action,

$$u(k)=k_c\cdot e(k) \tag{3.18}$$

$e(k)$ is the error given by,

$$e(k)=h_{sp}-h(k) \tag{3.19}$$

where k is the k-th sampling time.

The symbolic representation of a rule for a P-like FKBC is given as,

if e is (property symbol)

then u is (property symbol).

This type of FKBC gives offset in the controlled variable. Thus the integrating factor is needed to eliminate the offset.

PI- like FKBC:

A conventional PI-controller uses an analytical expression of the following form to compute the control action,

$$u(t)=k_c\cdot e(t)+\frac{1}{\tau_I}\cdot\int_0^t e\cdot dt \tag{3.20}$$

Differentiating this equation with respect to time we get,

$$\frac{du}{dt}=k_c\cdot\frac{de}{dt}+\frac{1}{\tau_I}\cdot e \tag{3.21}$$

The discrete time version of this equation can be written as,

$$\frac{\Delta u(k)}{\Delta t}=k_c\cdot\frac{\Delta e(k)}{\Delta t}+\frac{1}{\tau_I}\cdot e(k) \tag{3.22}$$

Where $\Delta u(k)$ is change-of-control output given by,

$$\Delta u(k)=u(k)-u(k-1) \tag{3.23}$$

$\Delta e(k)$ is the change of error given by,

$$\Delta e(k)=e(k)-e(k-1) \tag{3.24}$$

Also dh/dt the rate of change in height is given by,

$$\frac{\Delta h(k)}{\Delta t}=-\frac{\Delta e(k)}{\Delta t} \tag{3.25}$$

where k is the k-th sampling time.

The PI-like FKBC consists of rules of the form :

if e is (property symbol) *and* Δe is (property symbol) *then* Δu (property symbol) instead of analytical expression defining the conventional PI-controller. The presence of additional term Δe acts as integrating parameter, which eliminates the offset.

2. Choice of a TermSet:

Termset is a property symbol expressed linguistically like *no change*, *small positive*, *large negative*. These are called as linguistic values, members of the termset, which are in the form of value magnitude, and value sign also called as *tuple form*. The meaning of tuple in the case of a PI-like FKBC can be summarized as follows. Linguistic values of *e* with a *positive sign* mean that the current process output $h(k)$ has a value *below setpoint* h_{sp}. On the other hand, linguistic values of *e* with a *negative sign* mean that the current process output is *above the setpoint*. Similarly signs are defined for other linguistic variables. In above cases if Δh is used instead of Δe then signs will reverse. For the present study of liquid level control Δh is used. The following matrix explains the set of rules and describes the operational meaning of *if-then* rules. The cell defined by the intersection of the first row and the first column represents a rule such as,

if $\Delta h(k)$ is *Fs neg.* and $e(k)$ is *Lg neg.*

then $\Delta u(k)$ is *No change*

these rules are shown in Table 3.1 in matrix form.

AND		**Error, e**				
		Lg neg.	**Sm neg.**	**None**	**Sm pos.**	**Lg pos.**
Δh	**Fs neg.**	*No change* (1)	*Sm pos.* (1)	*Sm pos.* (3)	*Lg pos.* (3)	*Lg pos* (3)
	Sl neg.	*Sm neg.* (1)	*No change* (0)	*Sm pos.* (0)	*Sm pos.* (0)	*Lg pos.* (3)
	Zero	*Sm neg.* (1)	*Sm neg.* (0)	*No change* (0)	*Sm pos.* (0)	*Sm pos.* (2)
	Sl pos.	*Lg neg.* (4)	*Sm neg.* (0)	*Sm neg.* (0)	*No change* (0)	*Sm pos.* (2)
	Fs pos.	*Lg neg.* (4)	*Lg neg.* (4)	*Sm neg.* (4)	*Sm neg.* (2)	*No change* (2)

Table 3.1: Matrix showing different groups of decisions.

The set of rules is divided in the following five groups:

Group 0: In this group both Δh and *e* are small or zero (positive or negative). This means that the process output variable *h* has deviated from the set point but is still close to it. Then the amount of change $\Delta u(k)$, in the previous control output $u(k-1)$ is small or zero in magnitude.

Group 1: For this group of rules *e* is small or large negative, which implies that, $h(k)$ is above the setpoint. At the same time $\Delta h(k)$ is negative means that $h(k)$ is moving towards setpoint. Then the amount of change $\Delta u(k)$, in the previous control output $u(k-1)$ is small or zero in magnitude.

Group 2: For this group of rules *e* is small or large positive, which implies that, $h(k)$ is below the setpoint. At the same time $\Delta h(k)$ is positive means that $h(k)$ is moving towards setpoint. Then the amount of change $\Delta u(k)$, in the previous control output $u(k-1)$ is small or zero in magnitude.

Group 3: For this group of rules *e* is small or large positive, which implies that, $h(k)$ is significantly below the setpoint. At the same time $\Delta h(k)$ is negative means that $h(k)$ is moving towards setpoint. Then the amount of change $\Delta u(k)$, in the previous control output $u(k-1)$ is small or large in magnitude.

Group 4: For this group of rules *e* is small or large negative, which implies that, $h(k)$ is significantly above the setpoint. At the same time $\Delta h(k)$ is positive means that $h(k)$ is moving towards setpoint. Then the amount of change $\Delta u(k)$, in the previous control output $u(k-1)$ is small or large in magnitude.

3. Derivation of rules:

The most widely used approach is based on the derivation of rules from the experience-based knowledge of the process operator and/or control engineer. The approach is realized using two types of techniques.

i) An introspective verbalization of experience-based knowledge.

ii) Interrogation of a process operator and/or control engineer using a carefully organized questionnaire.

In the case of PI-like FKBC a closed loop response of the process under control is studied. This response is composed of error and change in error from which set of rules is prepared.

3.4.2 Data Base:

The design parameters involved in the construction of the database are choice of membership functions and choice of scaling factors. Scaling factors are required for input normlization and output normalization.

Choice of membership functions:

The linguistic values taken by the variables in the rule-*antecedent* (*if*-part) and rule-consequent (*then*-part), and symbolic representation of the rules are required for qualitative analysis. In membership functions the meaning of each linguistic value from the term set represented on the respective domains $e, \Delta e$ and Δu . For computational efficiency, efficient use of memory, and performance analysis needs, a uniform representation of the membership functions required. This uniform representation can be achieved by employing membership functions with uniform shape (see figure 3.7) and parametric, functional definition.

To establish standard membership functions, following steps are followed. Before establishing membership functions the dynamic behavior of linguistic variables is studied.

1. The typical value that best fits the linguistic meaning of the term and yields the membership degree $\mu = 1$ for each term is defined.
2. For each term at the typical values of neighboring terms membership degree is set to $\mu = 0$.
3. Then point $\mu = 1$ is connected by straight line with point $\mu = 0$ for all inner terms creating a triangular membership functions.
4. The extreme ends rightmost and leftmost part fall under membership degree $\mu = 1$ creating Z or S- shaped membership functions.

It is easy to seen that parametric functional description of the triangular shaped membership function is the most economic one.

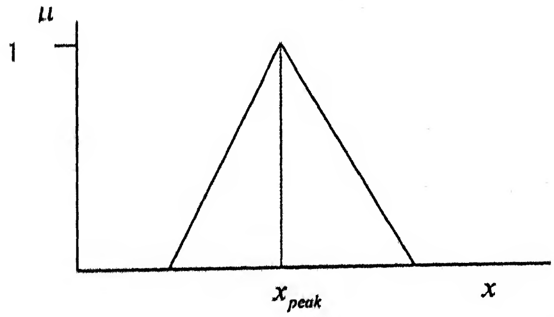
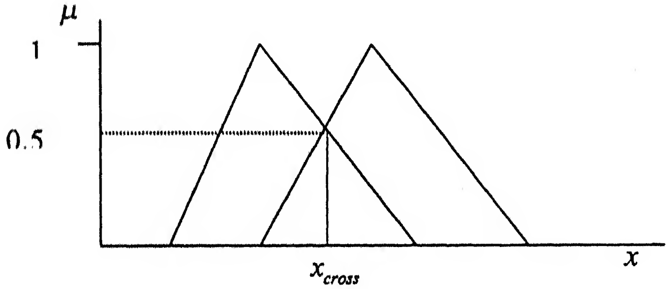


Figure 3.8: The peak value of a triangular membership function ($\mu = 1$).

Figure 3.8 shows the peak value of triangular membership functions. For trapezoidal membership functions the peak value is an interval. The left width is distance from x_{peak} to point where $\mu = 0$ at the left; similarly width is defined for right. If the two widths are equal then membership function is symmetrical otherwise unsymmetrical.

If there are two membership functions representing the meaning of a linguistic variable then there is a cross-point level and cross point ratio. A cross-point level is defined by degree of membership at cross point of two membership functions. Cross-point ratio is the number of cross points between two membership functions. The cross-point level should be greater than zero which means that every crisp value of error belongs to atleast one membership function with degree of membership strictly greater than zero, [Driankov et al. (1993)]. The typical values of cross-point level are 0.5 and that of cross-point ratio are 1, then this provides for significantly less overshoot, faster-rise time and less undershoot. Figure 3.9 explainscross-point level and cross-point ratio.



Cross-point level = 0.5

Cross-point ratio=1

Figure 3.9: Crosspoints and cross-point level of triangular membership functions.

3.4.3 Block Diagrams for fuzzy-PI control:

The following Figures 3.10(a), Figure 3.10(b) and Figure 3.10(c) show the block diagram developed for implementing fuzzy- PI control, see Appendix D. The motorized valve control algorithm is same as shown figure. Block diagram 3.10(a) shows calculation of time interval. This time interval is then passed to next sequence to VI: *fuzzy control.* The fuzzy controller gets database of control action to be taken from VI: *load control.*

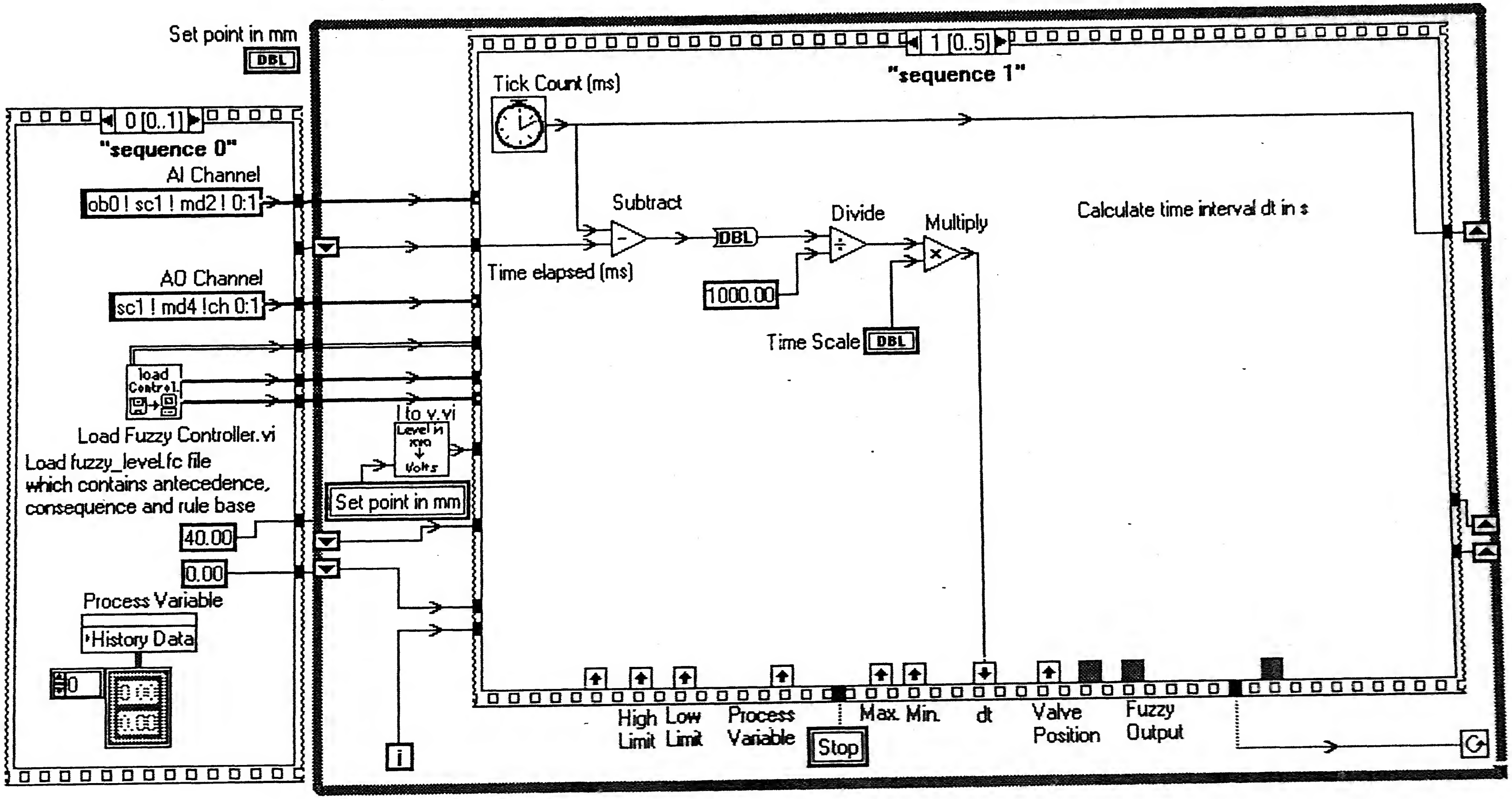


Figure 3.10(a): Block Diagram to calculate time interval.

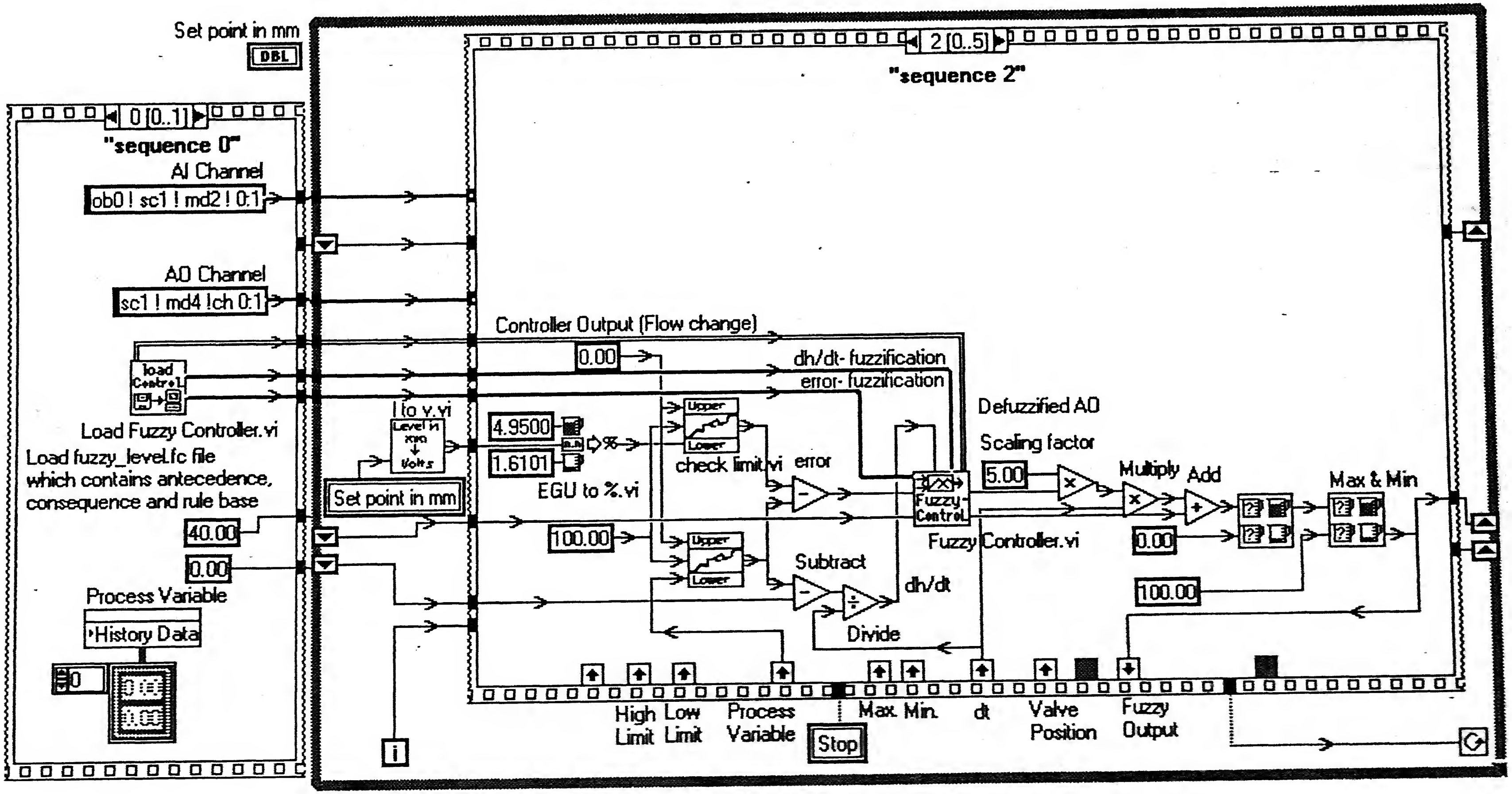


Figure 3.10(b): Block Diagram to calculate fuzzy controller output.

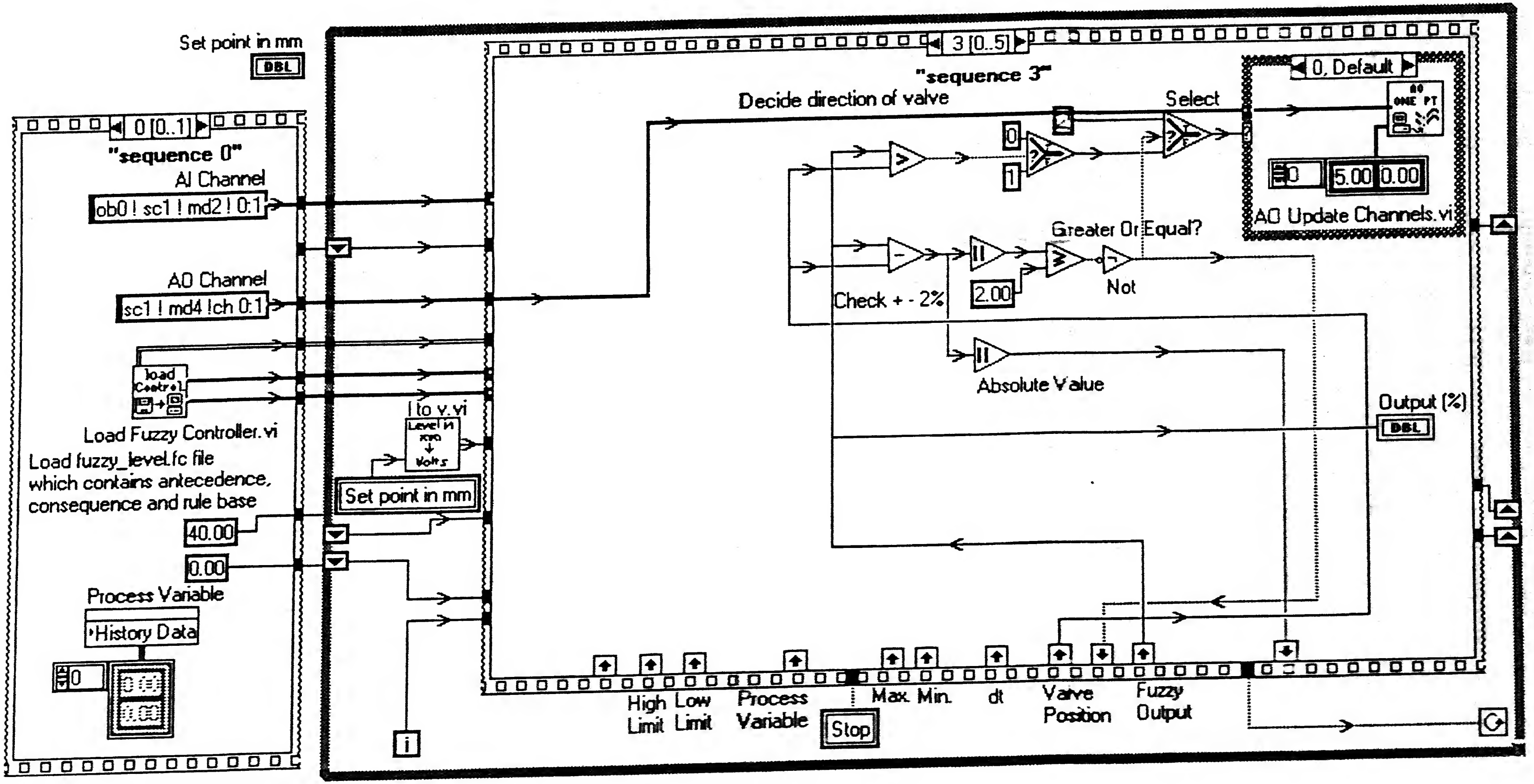


Figure 3.10(c): Block Diagram to decide direction of the valve.

Chapter 4
Results And Discussions

In this chapter the results of the experiment carried on liquid level set up using different algorithms is presented.

4.1 PI-control tuning using open-loop response:

Figure 4.1 shows the process reaction curve for liquid level. The nature of the curve depicts characteristics of the capacitive and integrating process. The level goes on building and requires theoretically infinite time to come to equilibrium; the system is also called as self-regulating system. The small positive or negative step changes thus cause the tank to overflow or run dry.

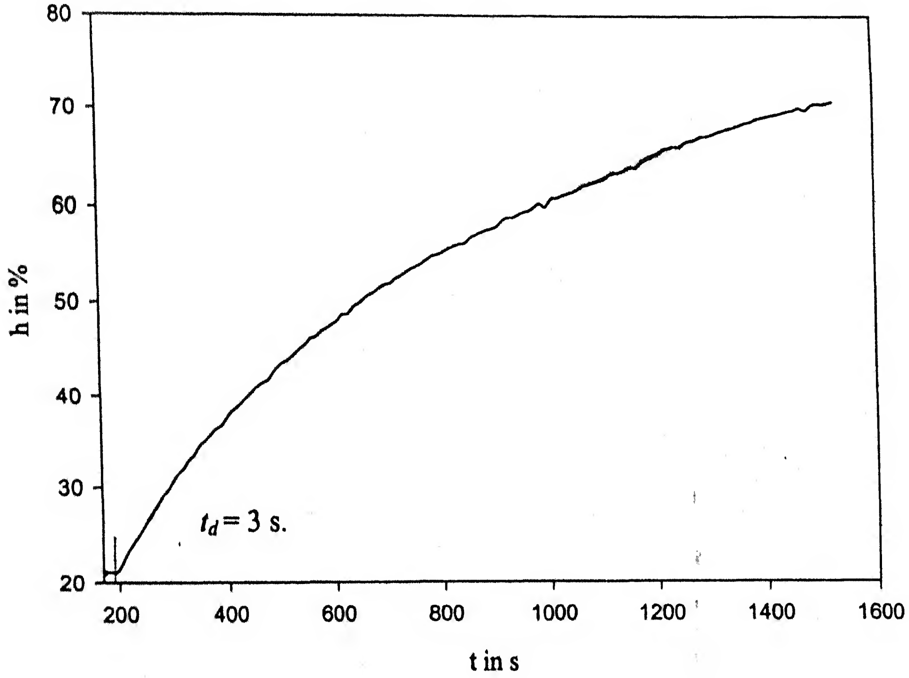


Figure 4.1: Process reaction curve of liquid level.

The PI-parameters are found from open-loop response using Cohen-Coon settings as k_c= 9.053, τ_I = 0.1645 min. The sampling time is chosen as 0.1 t_d. When sampling time is 1s or above response of liquid level found to be oscillating with more overshoot and undershoot. For a sampling time lesser than 0.1 t_d, no further improvement in response observed. Figure 4.2(a) shows the liquid level responding under PI-control.

The disturbance is given through set point changes. The process variable seems to be giving rise to overshoot and undershoot. Figure 4.2(b) shows corresponding valve position. Valve position is also seems to be oscillating. Figure 4.2(c) shows the change in valve position. The sudden jump in valve position is observed at setpoint changes. The performance index is defined as,

$$\text{P.I.(e)} = \sum_{0}^{t} \left(h_{SP} - h\right)^2 \tag{4.1}$$

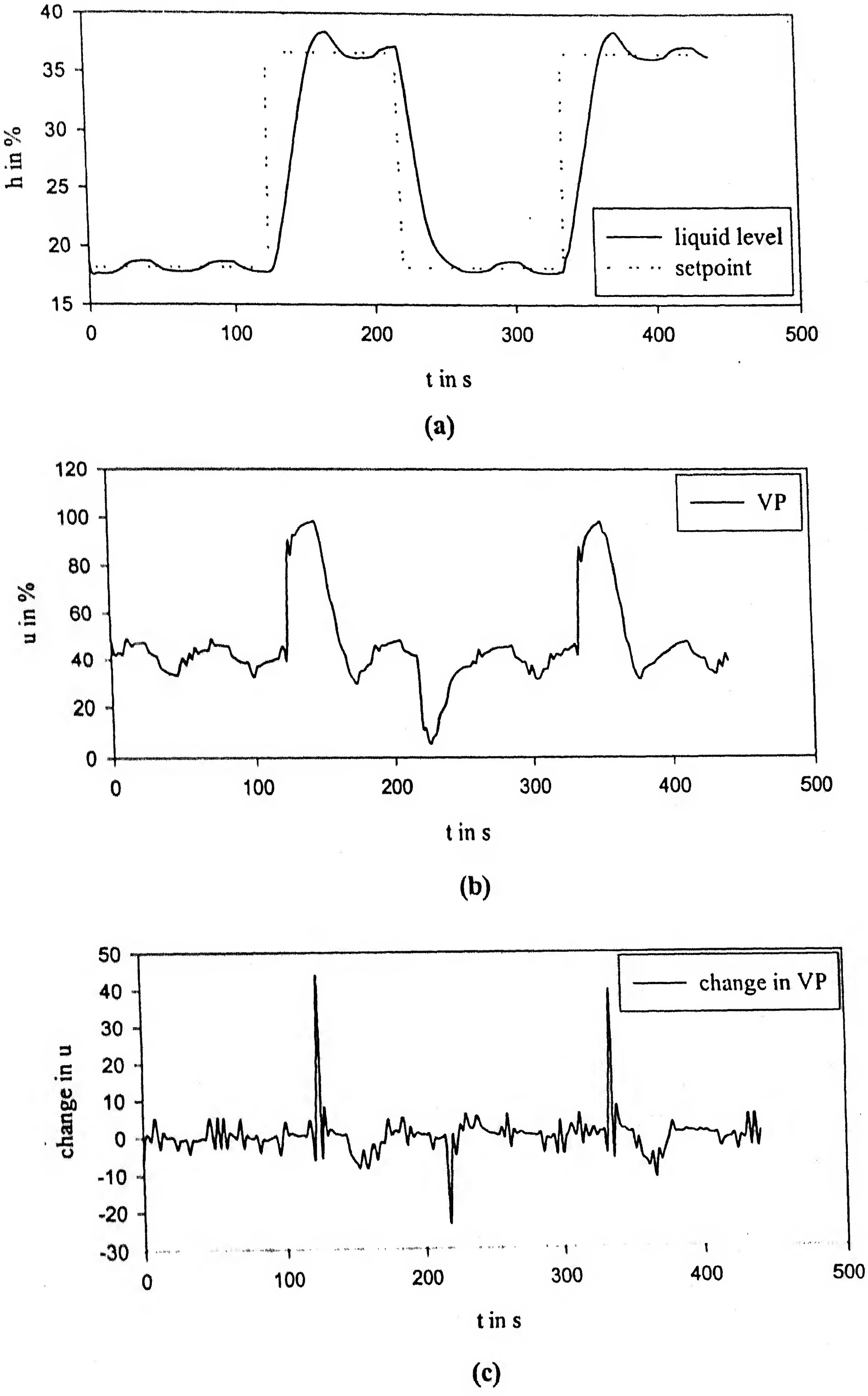
Another performance index based on change in u is defined as,

$$\text{P.I.}(\Delta u) = \sum_{0}^{t} \left(\Delta u\right)^2 \tag{4.2}$$

The performance index found for PI-control based on Cohen-Coon settings are P.I.(e) = 5492.787 and P.I.(Δu) = 3703.213.

4.2 Autotuning PI-controller:

Figure 4.3(a) shows the Autotuning PI control response of liquid level. The oscillations in the liquid level are reduced significantly. The zone between dashed lines shows setpoint relay experiment. The process variable i.e. liquid level is having sustained oscillations during the experiment. The relay amplitude used is 2% of the setpoint range. The basic requirement for autotuning procedure is to keep the valve moving between saturation limits i.e. valve should not touch the extreme limits. Since valve has a limiting position fully closed or fully open, it's not possible to determine the amplitude of the oscillation. In the present system relay amplitude of 3% and above causes valve to touch the saturation limits, hence a value of 2% is used. Figure 4.3(b) shows the valve position under Autotuning PI control. The valve seems to be behaving smoothly compared to Cohen-Coon settings based PI control. Figure 4.3(c) shows the change in valve position. It can be seen that sudden jerks in the movement of valve still persists. The PI-parameters obtained here are k_c=18.037, τ_I=0.2955 min. There is increase in proportional gain and integral time compared to open loop method. The performance index for the Autotuning-PI control is found to be P.I.(e) = 6345.796 and P.I. (Δu) = 3903.574 which is considerably more than previous.



(a)

(b)

(c)

Figure 4.2: Plots showing (a) liquid level (b) valve position (c) change in valve position under PI based on Cohen-Coon settings.

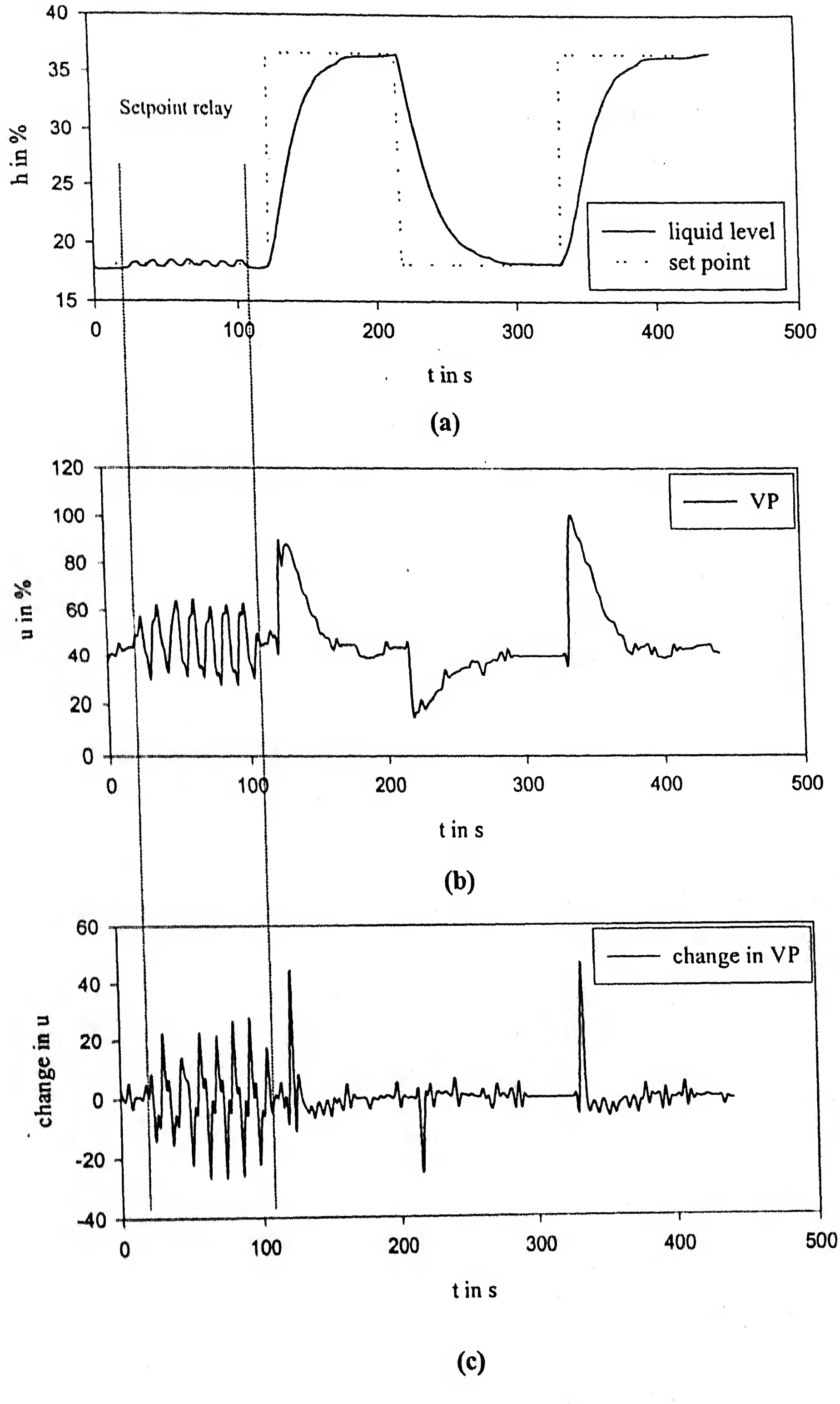


Figure 4.3: Plots showing response of (a) liquid level (b) valve position (c) change in valve position under Autotuning PI control.

4.3 Gain-Scheduling PI-control:

The liquid level sensor used in the present system shows a nonlinear behavior at above the 70% of process variable range, see Appendix B. Thus the system autotuned at two different operating points in 0-70% range and 70-100% range. The autotuning is done by setpoint relay method. Thus the two sets of PI-controller parameters are obtained at two different ranges of process variable. k_c= 18.037, τ_I= 0.2955 min., for 0 to 70 % range and k_c= 23.4864, τ_I= 0.1859 min., for 70 to 100 % range. The process gain has increased and integral time is reduced for second set of parameters. Figure 4.4 shows the closed-loop response under gain scheduling and autotuning PI control. Comparing with Figures 4.2 and 4.3 a typical process characteristic can be seen. When a small set point change is given, valve has moved at fully open position and response is faster. Then second step change of big magnitude is given. It can be seen that though the valve had moved to fully open position process variable has taken considerable time to reach to the desired value, i. e. The rate of rise of liquid level is decreasing as liquid level is increasing. Here a comparison is made between Gain Scheduling and Autotuning PI control. The performance index P.I.(e) for former is found to be $1.958*10^5$ and for later it is found to be $2.082*10^5$, also P.I.(Δu) is found to be 6706.021 and 6666.991. There is a slight improvement in the performance index based on error.

From Figure 4.4 it can be observed that in both cases the oscillatory behavior in the process variable is significantly low. Figure 4.5 show corresponding valve position. In autotuning valve is less subjected to sudden movement than gain scheduling. Figure 4.6 shows the change in valve position. In gain scheduling valve movement slightly greater than autotuning. Table 4.1 shows parameters obtained.

Parameters	Gain Scheduling	Autotuned PI
k_c	18.037 for 0-70% range of process variable 23.4864 for 70-100% range of process variable	18.037 for 0-100% range of process variable
τ_I , min.	0.2955 for range 0-70% 0.1859 for range 70-100%	0.2955 for 0-100% range of process variable
P.I.(e)	$1.958*10^5$	$2.082*10^5$
P.I.(Δu)	6706.021	6666.991

Table 4.1: Parameters obtained for Gain Scheduling and Autotuning PI control.

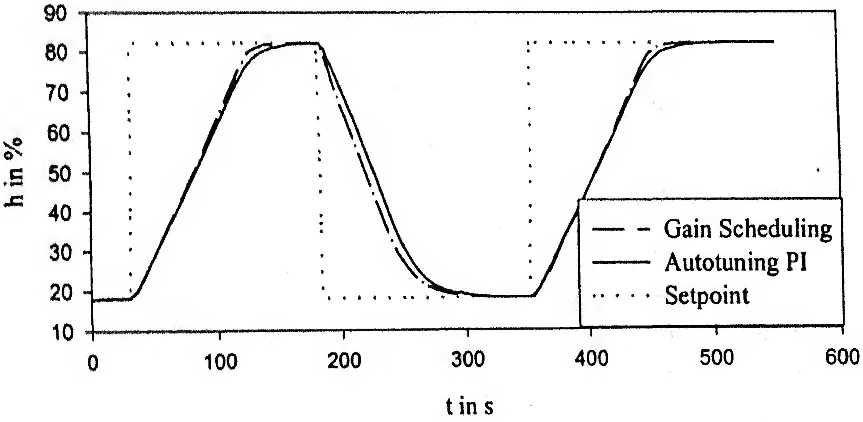


Figure 4.4: Plots showing response of liquid level under Gain Scheduling and Autotuning PI control.

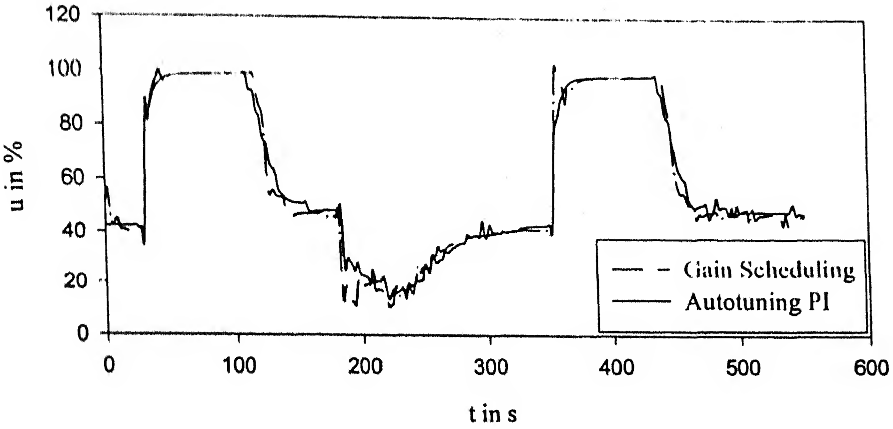


Figure 4.5: Plots showing response of valve under Gain Scheduling and Autotuning PI control.

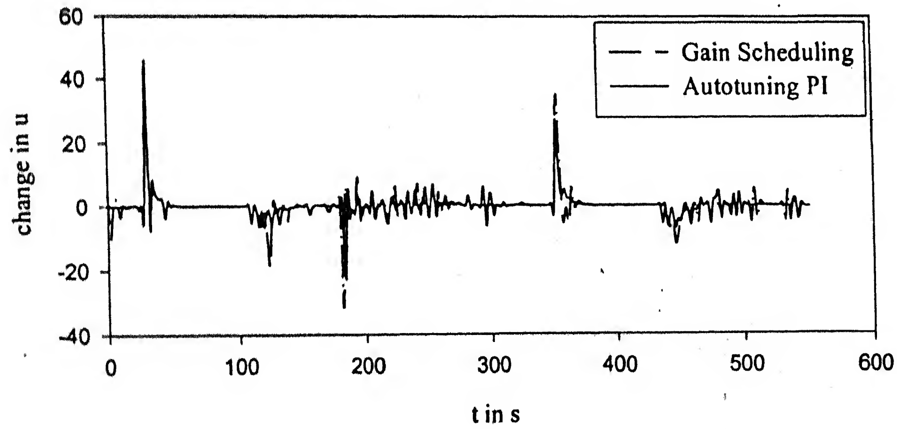


Figure 4.6: Plots showing response of change in valve position under Gain Scheduling and Autotuning PI control.

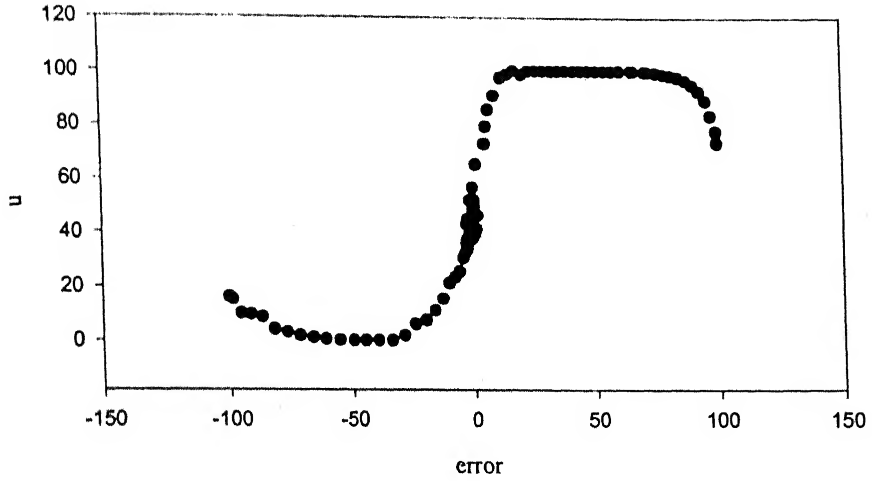
4.4 Fuzzy-PI Control:

In the present section results are presented according to steps followed.

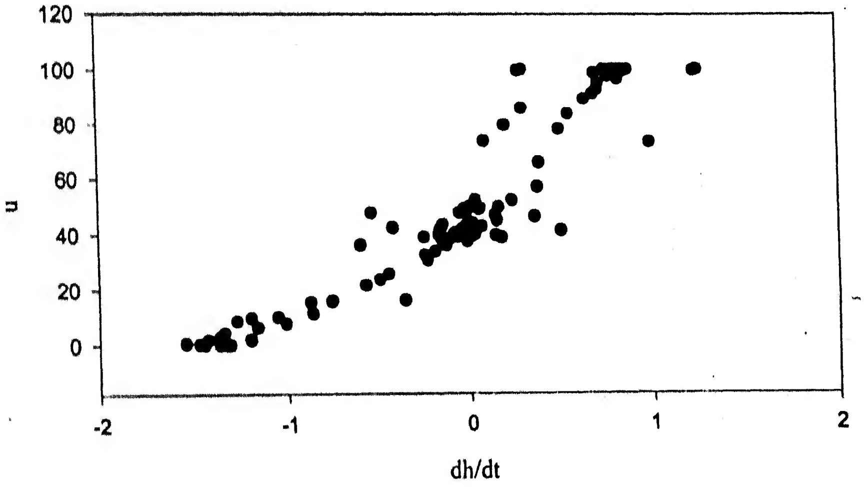
4.4.1 Physics of the process:

The first step taken in constructing the fuzzy-PI control is to understand the physics of the process. Thus an experimental data of process under closed loop is observed. Figures 4.7 to 4.10 give the idea of the physics of the system involved. These figures show how the u or Δu are changing with error or rate of change in height. In the present system error and dh/dt are chosen as the linguistic terms of the antecedence part and change in flow, Δu , as the linguistic variable of the consequence part.

The figures contain unfiltered and filtered data. It can be seen that its bit of difficult to understand the nature of the process by observing unfiltered data except for plots of u. Unfiltered data is one which has came form process. This non-uniform scattering of data can be attributed to the noise and tendency of the process variable to get jagged at zero error. Thus to eliminate the noise and acquire meaningful representation of the data filtering is done. There are different types of filters available in LabVIEW like Butterworth filter, Chebychev filter, FIR Windowed filter. The filtered data shown in Figure 4.8(a) is compared with less noisy data points shown in Figure 4.7(a). The filtered data shown in Figure 4.8(b) is compared with Figure 4.7(b), which is having more scatter. The filtered data shown in Figure 4.10(a) is compared with less scattered data points shown in Figure 4.9(a). The filtered data shown in Figure 4.10(b) is compared significantly scattered data points shown in Figure 4.9(b). The filtered data is acceptable provided that these data points do represent similar behavior of the process as depicted by original experimental data. Thus a filter is selected which will closely follow the nature of the original data points. Thus among the above mentioned filters Windowed filter is found to be suitable for filtering data, see Appendix E. The Low pass filter option is used since it passes low frequencies and attenuates high frequencies (data points where very small changes are occurring). The default values of sampling frequency as 1 and low cut off frequency as 0.125 are used. Thus the data is neatly arranged and manipulated so those regions where significant changes are taking place are identified.

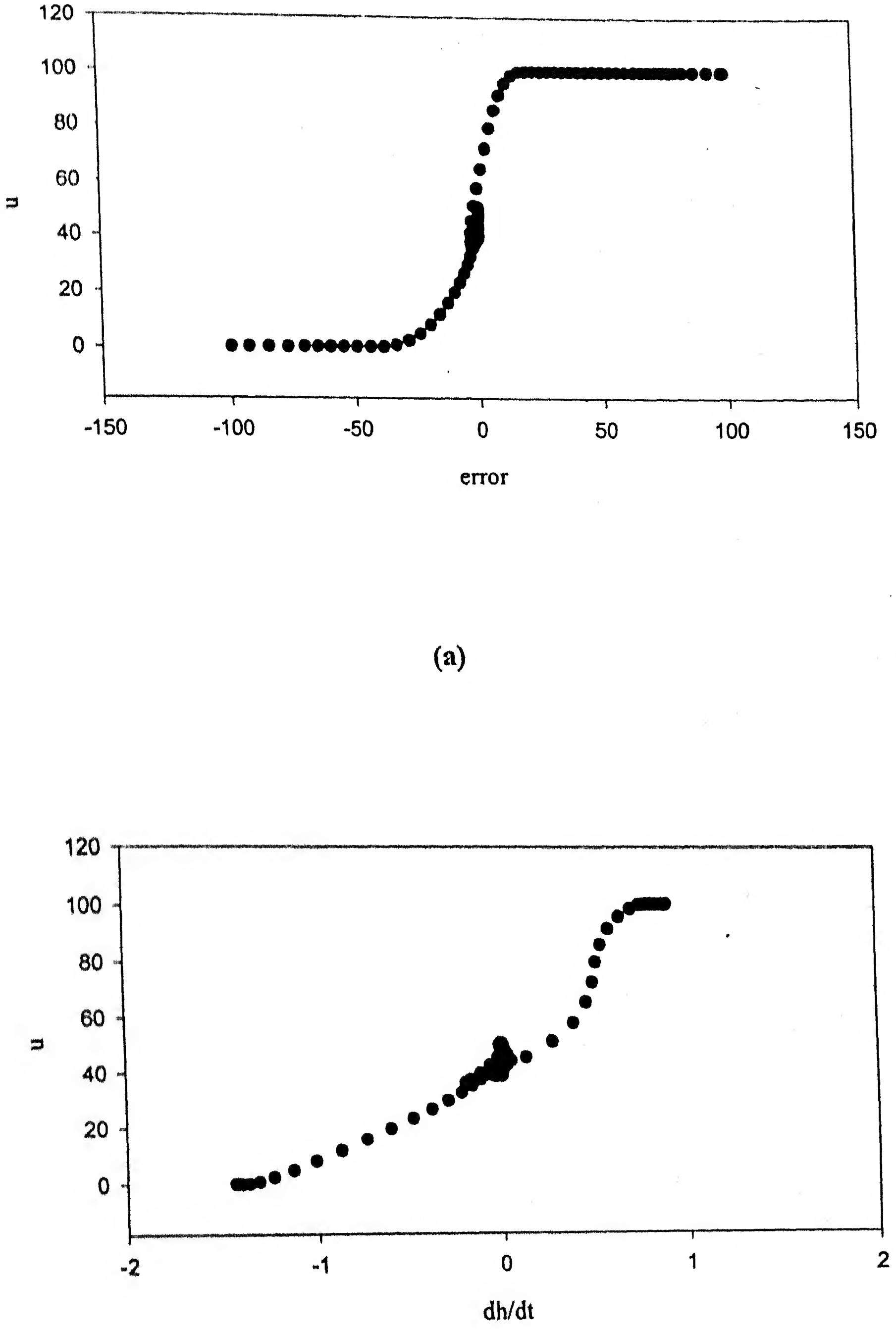


(a)



(b)

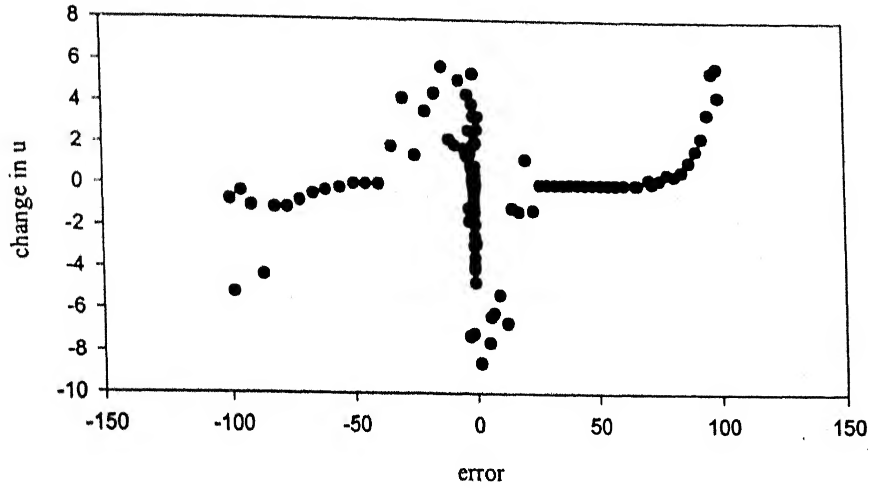
Figure 4.7: Plot showing *u* (flow) varying with (a)error (b) *dh/dt* (Unfiltered data).



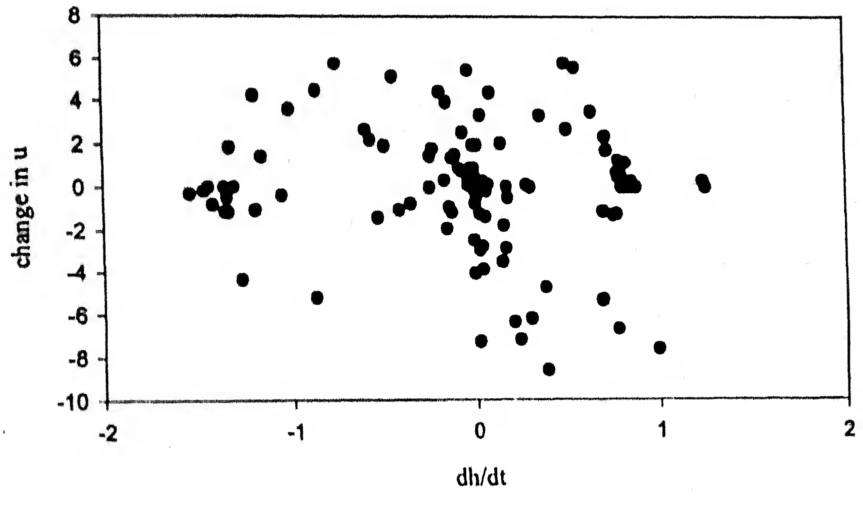
(a)

(b)

Figure 4.8: Plot showing *u* (flow) varying with (a)error (b) *dh/dt* (Filtered data).

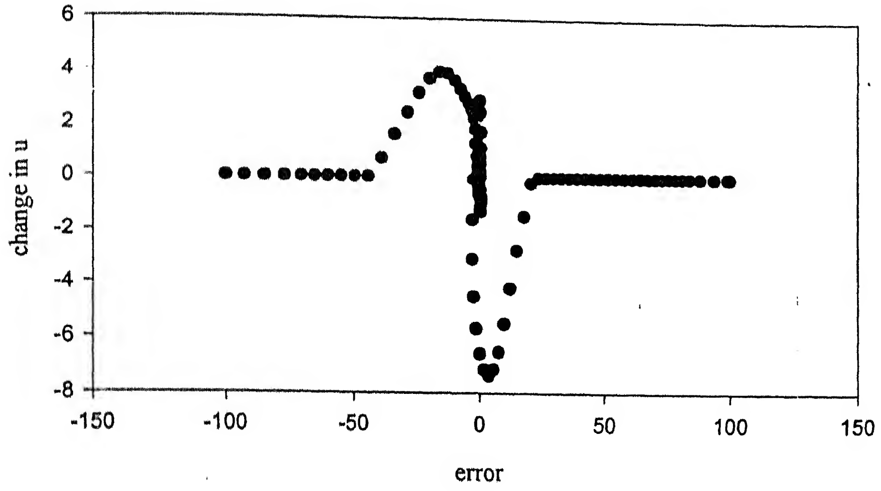


(a)

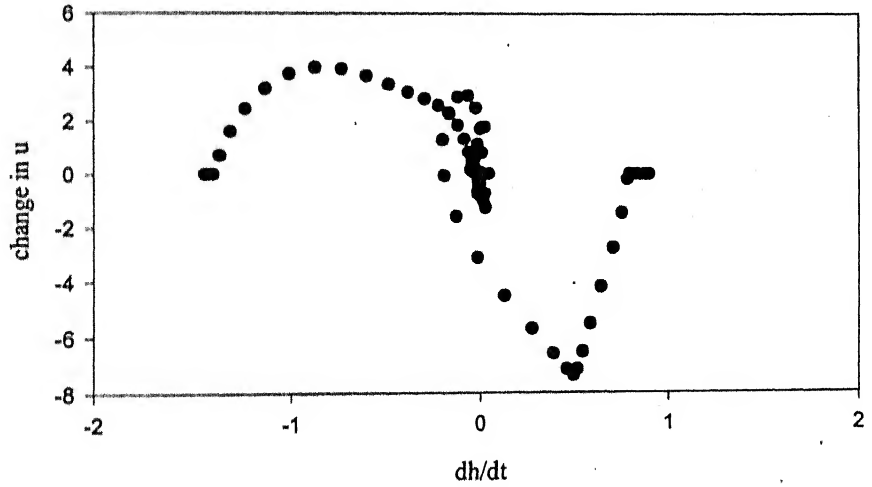


(b)

Figure 4.9: Plot showing change in *u* (flow change) varying with (a)error (b) *dh/dt* (Unfiltered data).



(a)



(b)

Figure 4.10: Plot showing change in *u* (flow change) varying (a)error (b) *dh/dt* (Filtered data).

4.4.2 Identifying fuzzy regions:

Figure 4.11 shows the dynamic response of flow rate *u*. Figure 4.12 shows dynamic responses of linguistic variables. To construct membership functions the regions are demarcated as shown by horizontal dashed lines as shown in Figure 4.12. From where a typical value of membership function is decided. The membership functions constructed are shown in Figure 4.13 to Figure 4.15. By observing dynamic plots of linguistic variables, also by observing the relation between linguistic variables physical values of the variables are mapped in fuzzy zone. Scaling factor of 5 is used for flow change. This scaling factor takes account of the sudden disturbances to the process. If scaling factor is not incorporated response becomes oscillatory. The time scale factor is also important for getting fast responses. Time scale factor of 10 is used here. The higher time scale factor causes overshoot and lower time scale causes low-rise time.

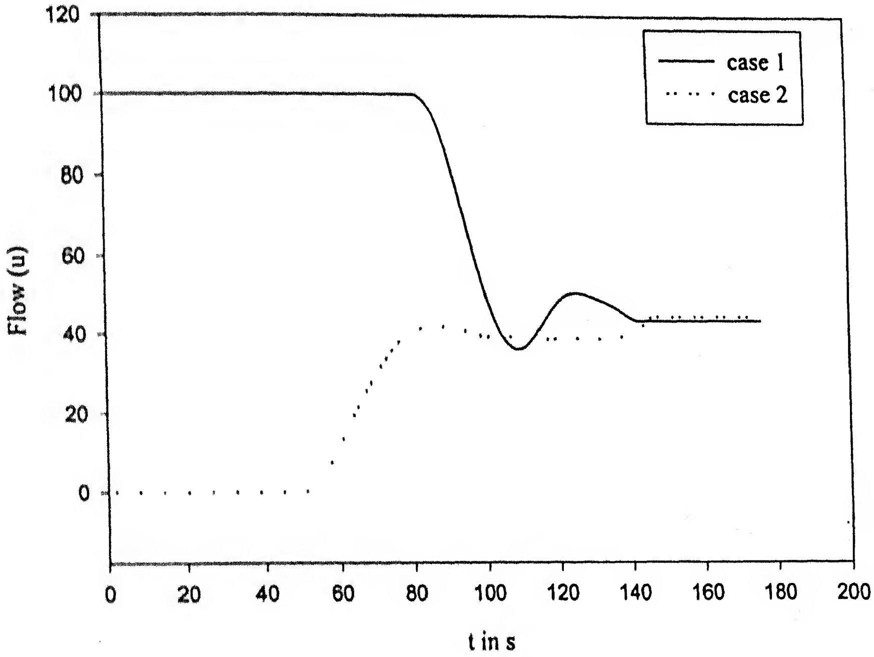


Figure 4.11: Plot showing dynamic behavior of flow rate (*u*).

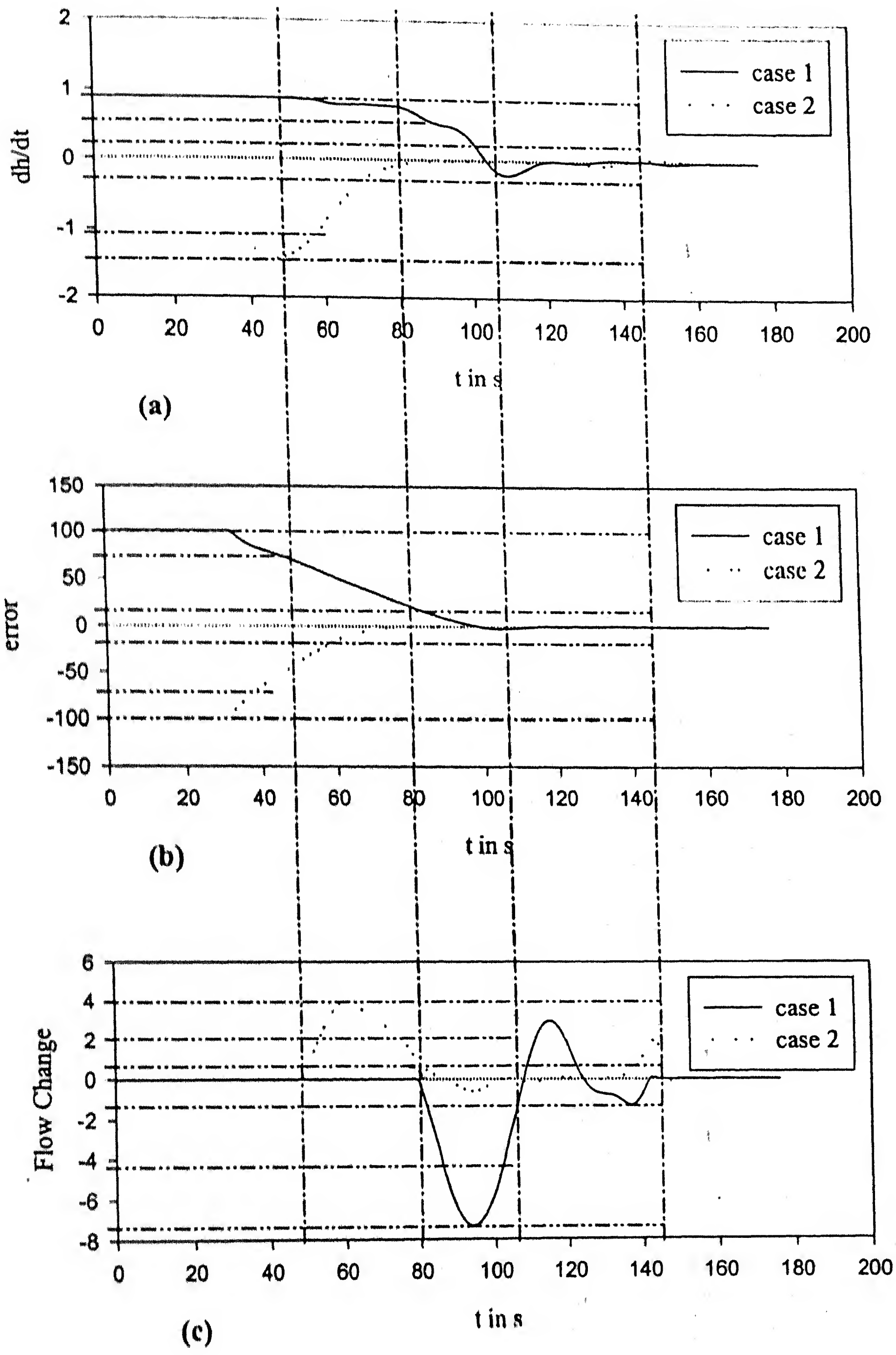


Figure 4.12: Dynamic behavior of change in (a) liquid level (b) error and (c) flow change.

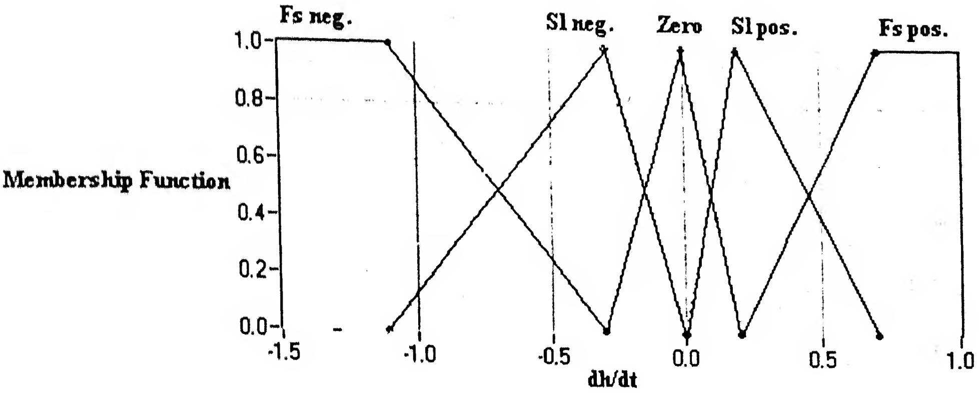


Figure 4.13: Membership function for change in liquid level with time

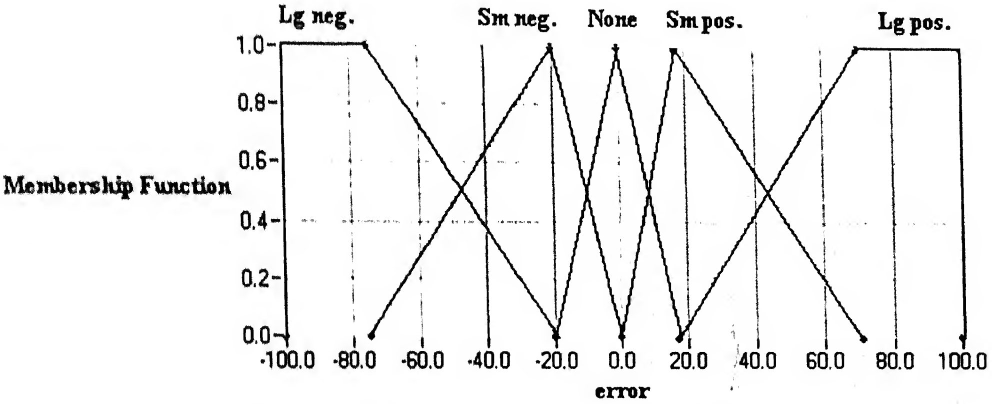


Figure 4.14: Membership function for error

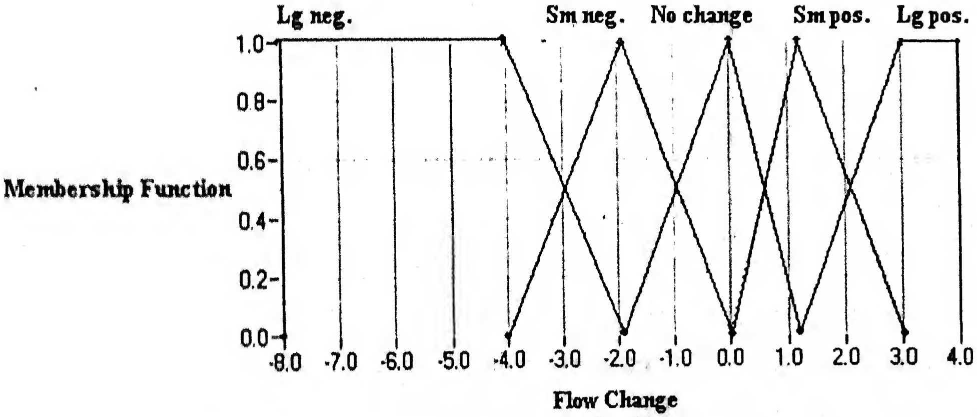


Figure 4.15: Membership function for change in flow

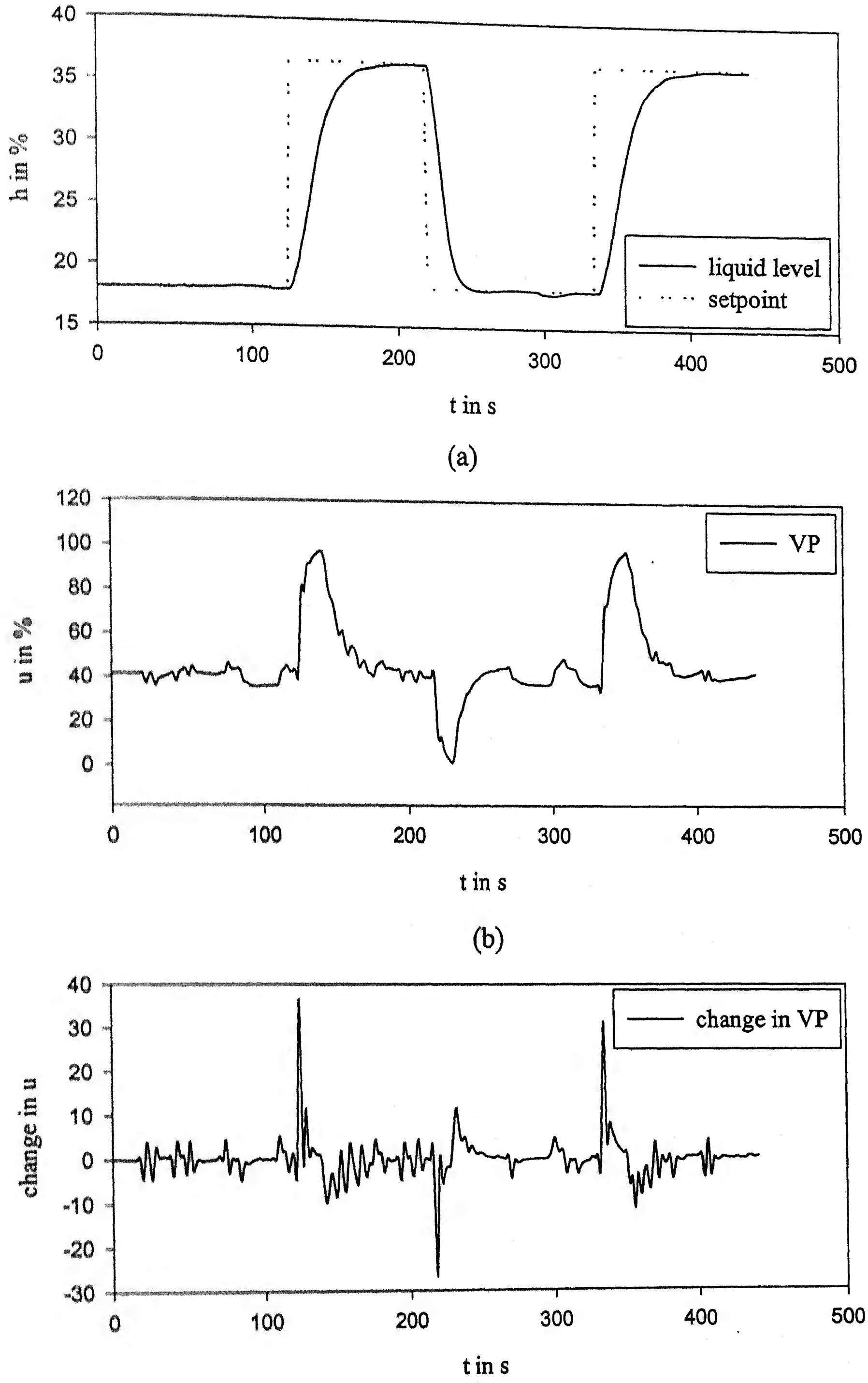
4.4.3 Rule Base:

Based on dynamic plot rule base is prepared. The vertical dotted lines shown in Figure 4.10 depict the zone where changes in linguistic variables are taking place with time. The horizontal lines show the region where linguistic term like *neg.*, *pos.* are applied. The middle dotted line and corresponding neighboring lines show the no change zone. Upper to the no change zone lies positive zone and below lies negative zone.. The extreme zone between dotted dashed lines is termed as *Lg* for error and flow change while for slope it is *Fs*. The zone between extreme and middle is termed as *Sm* for error and flow change while *Sl* is termed for slope. Figure 4.11 shows dynamic response of *u*, is also taken into account while deriving rules. Table 4.2 shows the 21-rule base mapped in to 25 zones.

AND		**Error, e**				
		Lg neg.	**Sm neg.**	**None**	**Sm pos.**	**Lg pos.**
dh/dt	**Fs neg.**	*No change*	*Sm pos.*	*Sm pos.*	*Lg pos.*	*Lg pos.*
	Sm neg.	*Sm neg.*	*No change*	*Sm pos.*	*Sm pos.*	*Lg pos.*
	Zero	*Sm neg.*	*Sm neg.*	*No change*	*Sm pos.*	*Sm pos.*
	Sm pos.	*Lg neg.*	*Sm neg.*	*Sm neg.*	*No change*	*Sm pos.*
	Lg pos.	*Lg neg.*	*Lg neg.*	*Sm neg.*	*Sm neg.*	*No change*

Table 4.2: Matrix showing database of 21-rules.

Figure 4.16(a) shows closed loop response of liquid level under fuzzy-PI control. The oscillations in this type of controller are significantly reduced. Figure 4.16(b) shows the behavior of final control element under the fuzzy-PI control. The valve movement is in steps and in linear fashion, which helps in avoiding sudden jerks. The reason of linear movement can been seen from Figure 4.15 where output is calculated region



(a)

(b)

(c)

Figure 4.16: Plots showing response of (a) liquid level (b) valve position (c) change in valve position under Fuzzy-PI control.

wise i.e. from one region to other. Figure 4.16(c) shows the change in valve position. The valve movement is occurred more than the autotuning PI control and Cohen-Coon settings based PI control. The performance index is found to be P.I.(e) = 6089.554, which lies in between the Cohen-Coon settings based PI-control and Autotuning PI-control. But the oscillations in the process variable are totally reduced. There is less difference in performance index of Fuzzy-PI and Cohen-Coon settings based PI compared to Autotuned PI based on Zigler-Nichols settings. The performance index for valve movement is found to be P.I.(Δu) = 3327.079 which is less than all of the previous control strategies. Table 4.3 shows the comparison of parameters for three different control strategies applied on liquid level.

Parameters	**Cohen-Coon settings based PI control**	**Autotuned PI control**	**Fuzzy-PI control**
k_c	9.053	18.037	Related to membership function of error.
τ_I, **min.**	0.1645	0.2955	Related to membership function of dh/dt.
P.I.(e)	5492.787	6345.796	6089.554
P.I.(Δu)	3703.213	3903.574	3327.019

Table 4.3: Table showing parameters for different control strategies.

Chapter 5

Conclusions And Recommendations

A Virtual Instrument PC-based fuzzy control is developed for liquid-level control using Fuzzy logic toolkit available in LabVIEW. Autotuning and Gain Scheduling are studied using PID toolkit. Open loop method for determining PI parameters also studied. For implementing control strategies VI based algorithms for measurement of signals, motor control algorithms are developed. Thus by doing a comparative study of previous chapters following conclusions can be drawn:

- Just a superficial knowledge of electrical and instrumentation is required for PC-based control using Virtual Instrumentation. Installation of SCXI-1001, SCXI-1200, SCXI-1100, SCXI-1124 has been done and PC-interfacing has been thus done for acquiring liquid level, valve position signals and sending signal to the actuator.
- A VI for process control can be developed easily and different control strategies can be studied in a short period of time. Thus VIs for mesurement of input and output signals and motor control have been developed. Process reaction curve, autotuning, gain scheduling and fuzzy control have been implemented and studied.
- Performance of Autotuning and Gain-Scheduling control strategies found similar.
- Autotuning compared with Cohen-Coon settings is having less satisfactory performance in terms of performance index. Autotuning gains advantage only in avoiding overshoots and undershoots i.e. oscillations in process variable (liquid level).
- Fuzzy-PI control performance is found acceptable compared to Cohen-Coon settings based PI control and Autotuning PI control based on Zigler-Nichols settings.
- The valve movement is found to be less erratic in Fuzzy-PI control among all control strategies.
- For implementing fuzzy control only the experimental data is required to evolve only the rule database; detailed analytical modeling is not necessary.

Recommendations for future work:

- The control algorithms like fuzzy control, autotuning, gain scheduling, can be applied to laboratory experiments like temperature control flow control, pressure control, pH control and also to the bench scale experiments.
- Fuzzy control can be extended to construct adaptive control strategies.
- Performance index of the fuzzy control can be further improved by shifting of membership functions and/or creating more regions around most active zone. Scale factors for each linguistic variable also play important role in improving performance index.
- Connection between dead time and sampling time has to be explored properly for good performance of controller; the effect of dead time of the motorized valve on sampling time also should be taken into account.
- The current work is based on offline construction of membership functions. Scope is there for making it online by adjusting the coordinates.
- Fuzzy control is mainly useful where modeling of processes becomes very complex. Thus it can be applied to separation processes like bench scale liquid-liquid extraction where holdup control is very important.
- Different types of control algorithms like feedforward control, dead-time compensation can be created in VI and studied on different experiments.

Appendix A

Hardware

A.1 Installation and configuration of SCXI System:

To use the *parallel port* (LPT port) to control the *SCXI system*, then *SCXI-1200* is taken as *first module* in the *chassis*. The parallel port cable connects directly from the back of SCXI-1200 to the LPT port of the computer.

1.Insert the SCXI-1200 module into the first slot (closest to power switch) of the chassis and add other modules in the subsequent slots, making sure the *chassis* is *powered off* when inserting the modules.

2. Next, cable the module to the computer by connecting one of the 25-pin D SUB ends of the parallel port cable to the PC parallel port and the other end to the back of the SCXI-1200. Screw in the mounting screws on the connectors to establish a firm connection.

3. Power on the chassis.

4. Go to Start >> Control Panel >> Add new hardware. Click the "next" button until you reach list of hardware. Select *SCXI-1200* under *Data Acquisition Devices*. If Data Acquisition Devices does not show up, it is likely that NI-DAQ is not installed correctly.

5. In Windows 95/98, SCXI-1200 (added in step 4) appears listed under *Devices and Interfaces*. Double click on the "Devices and Interfaces" Right-click on SCXI-1200 and select *Properties*.

6. In the next window, Click on *Modify*.

7. From the pull down ring menu, select the parallel port (e.g. LPT1). Click OK and *Next* in the following window.

8. Set the operating mode, polarity/range and the accessory used and click "Finish." Now SCXI-1200 gets listed under the *Devices and Interfaces*.

9. Right-click on SCXI-1200 and select *Properties.*

10. Click on *Test Resources* to test your SCXI-1200. If SCXI-1200 fails the test, follow these *troubleshooting* tests.

10.1 Before testing your SCXI-1200, make sure that the chassis is turned on and the parallel port cable is properly secured on both ends.

10.2 Make sure that an LPT port is properly assigned to the SCXI-1200.

10.3 The LPT port should have an input/output range and interrupt request (IRQ) assigned. Disable any *DMA* (*Direct memory access*) for the LPT port. Make sure that the SCXI-1200 passes the test first. If the *SCXI-1200 fails*, follow the steps above once again.

10.4 If SCXI-1200 doesn't passes the test, check to make sure that other modules are plugged in correctly.

10.5 It could be a case of bad module or a bad chassis.

11. If SCXI-1200 passes the test, go back to the *Devices and Interfaces* window, right-click anywhere in the empty space, and select *Insert* from the list, select *SCXI-1001* to add the chassis.

12. The next window will show you the chassis ID and address. Accept the default and click *next.*

13. Select *Autodetect* and click *next.*

14. Select SCXI-1200 and click *next.* A list of modules that has been detected appears. Make sure that the chassis is turned on and all the modules are plugged in.

15. Right-click on each module and select properties. *Set all the properties* under the *General, Channel, and Accessory* tabs. To make settings like operating mode, gain and filter settings, terminal blocks etc, make sure that the settings made here match the jumper settings (if any) on the module/s. Set the properties of other modules in the same way.

16. After setting the properties of all the modules, return back to the main MAX (Measurement and Automation eXplorer) window. Now the configuration is complete. Right-click on SCXI-1000 chassis from the *Devices and Interfaces* list and select *Test*.

17. A message that *The chassis is verified* should appear.

A.2 Accessing Channels:

General format of the string is,

Obz! SCx ! Mdy ! a is used for AI channels (SCXI-1100),

SCx ! Mdy ! ch a is used for AO channels (SCXI-1124)

This will measure Channel a (a:b for scanning multiple channels) on the module in slot y of the chassis with ID x which is multiplexed into onboard channel z.

A.3 SCXI related errors and possible remedies:

Application Engineers from National Instruments answered the following questions.

Question-1:

Getting error –10403 when SCXI-1200 is accessed by specifying channel strings. Why? (*Error-10403*: *deviceSupportError*, the specified device does not support the requested action (the driver recognizes the device, but the action is inappropriate for the device)).

Answer-1:

First of all, let make sure you have installed your hardware correctly. We have a good *installation troubleshooting web page*, follow these *links*: www.ni.com >> Technical Support >> Select Data Acquisition from the first pull down menu and hit GO >> Select Installation and Configuration and hit GO >> *Select OS and DAQ board* and hit GO >> *Board Installation & Configuration*. Now that you have confirmed that everything is installed correctly, lets discuss how the SCXI-1200 works. In essence the *SCXI-1200 is a DAQ card*, but manufactured in the SCXI form factor. Hence, you communicate with the SCXI-1200 by the *device number* listed in *Measurement&Automation*. For example, if you want to do an *analog output* on channel 0 of the SCXI-1200 you would insert the device number in *AO config.vi* and insert 0 into the channel string. However, You need to use channel strings to communicate to other modules in the SCXI chassis. The general format of the channel string: OBz!SCx!MDy!a . This will measure Channel a (a:b for scanning multiple channels) on the module in slot y of the chassis with ID x which is multiplexed into onboard channel z. The general format of the *AO channel string* is SCx!MDy!ch a.

Note-1: To avoid error-10403, SCXI-1200 is accessed through channel string e.g. 0. For other SCXI modules use of channel string syntax is made.

Question-2: Chassis back plane fuse is blowing off frequently. Why? Is there any possibility of any component gone bad due to short-circuiting because of power fluctuations?

Answer-2: The specifications on the chassis power supply are 240 VAC +/- 10%. Therefore, it should be good up to 264 VAC. If the input voltage spike are exceeded this level, you may want to consider adding some power conditioning equipment. Perhaps, you could add a surge protector and/or uninterruptible power supply to accomplish the power conditioning for you. The *power entry fuse* for 240 VAČ input voltage is specified as *0.8 A*. You can order this fuse from us (NI part# 766106-01) or from Wickman (part# 19195-046). As for your last question, a voltage spike could damage this equipment, as it can damage any electronic equipment. First, test to see if the chassis verifies. Then sequentially test an input of each SCXI module, until you are satisfied that they are working also.

Note-2: To avoid any kind of damage to the chassis due to power fluctuations, power conditioning equipment like voltage stabilizer, UPS system is used.

Question-3:

Getting error-10440 when trying to *access ports* B or C of SCXI-1200, also getting error-10800 after running VI for several minutes using SCXI-1200 and SCXI-1100, why?

(*Error-10440*: *sysOwnedRsrcError*, the specified resource is owned by the driver and cannot be accessed or modified by the user.

Error-10800: timeOutError, the operation could not complete within the time limit.)

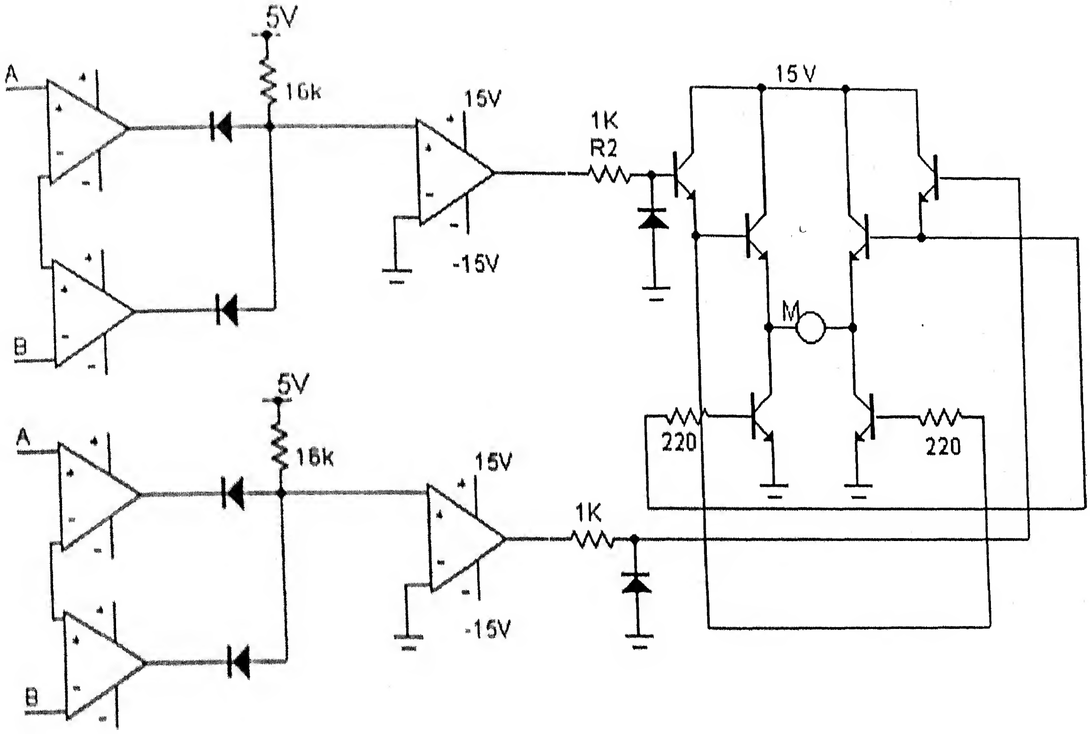
Answer-3:

I found some information on both of the errors that you are seeing with your SCXI setup. The *first issue* regarding the 10440 error is a new problem to NI-DAQ Judging from past customer issues, this error seems to occur when one tries to access the DIO ports after completing an AO operation. There are *two possible solutions* to this situation. First, you can make sure that you complete the DIO operations before you do any AO, which should stop this error from occurring. If that's not possible, then it might be best to *un-install NI-DAQ 6.7* and install NI-DAQ 6.6, which doesn't seem to have this problem. Just for your information, R&D is working on a fix for this for NI-DAQ 6.9.

I found some information regarding the second issue as well. A couple of our customers were experiencing similar difficulties (error 10800 after several minutes of running) in *environments of high humidity and/or hot temperatures*. This may not be related to sampling rate or buffer allocation. You can also try to add some more *loop delay time* to prevent the timeout error.

You will *not be able* to use all 24 digital lines on the SCXI-1200, unless it is the only module in your system. Certain lines on *ports B & C* are reserved for multi-module systems. To use all lines, assuming there are no other modules in the system, you must not configure the chassis in NI-DAQ configuration. Instead, just add the SCXI 1200 as a DAQ device and run the test panels. There may be some problems with this within NIDAQ 6.7 & 6.8 that will be corrected in NI-DAQ6.9. Be sure to uninstall your current version first.

Note-3: Error-10800 was a software driver problem. Thus, NI-DAQ-6.7 was uninstalled and NI-DAQ-6.6 was installed. Currently NI-DAQ-6.8.1 is installed and is working. The last paragraph is a very important piece of information and can be taken as warning to the users.



A	*B*	*Motion Of The Motor*
1	0	Forward motion
0	1	Reverse motion
1	1	No motion
0	0	No motion

Figure A.1: The circuit diagram (Driver Circuit) to run DC motor of motorized valve (Courtesy Mr. Tarun Gupta and Mr. Vikas Kataria both BTP (Electrical) students).

Appendix B

Calibration

The Figure B.1 Shows the calibration curve for liquid level and corresponding electrical signal. Table B.1 provides maximum and minimum voltages corresponding to position of valve fully open or fully closed.

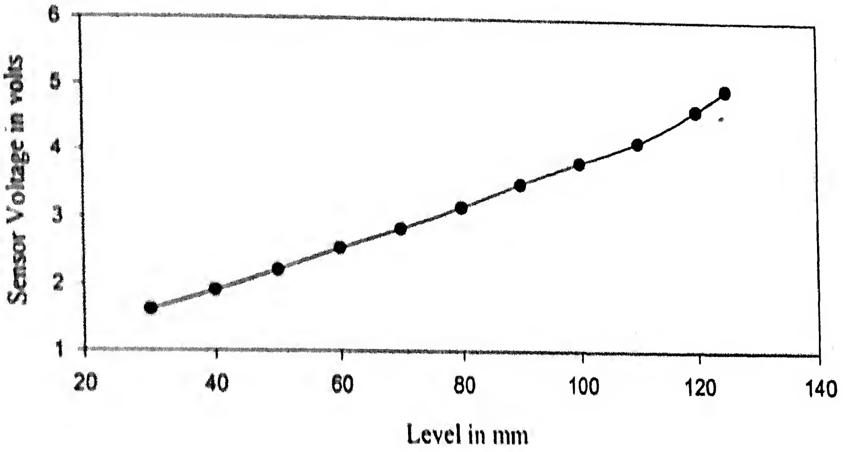


Figure B.1: Calibration curve for liquid level and corresponding sensor voltage.

Valve Position	Potentiometer Voltage in volts
Fully Open	3.42
Fully Closed	1.54

Table B.1: Potentiometer output for valve position.

Appendix C

List of SubVIs

C.1 List of subVIs for Autotuning and fuzzy logic control VIs:

Connector Pane	*Name*	*Location (File path)*
Level in mm ↓ Volts	**l to v.vi**	D:\mahesh\project\level control*l to v.vi*
Filter PV	**PV filter.vi**	*C:\LABVIEW\vi.lib\addons\PID\prctrl.llb\*PV filter.vi*
Volts ↓ Level in mm	**v to l.vi**	D:\mahesh\project\level control*v to l.vi*
AI ONE PT	**AI Sample Channels.vi**	*C:\LABVIEW\vi.lib\DAQ\1EASYIO.LLB\*AI Sample Channels.vi*
AO ONE PT	**AO Update Channels.vi**	*C:\LABVIEW\vi.lib\DAQ\1EASYIO.LLB\*AO Update Channels.vi*
PID AUTO	**PID with Autotuning.vi**	*C:\LABVIEW\vi.lib\addons\PID\autopid.llb\*PID with Autotuning.vi*
Upper Lower	**check limit.vi**	D:\mahesh\project\pid controls*check limit.vi*
12.3 TXT	**Write To Spreadsheet File.vi**	*C:\LABVIEW\vi.lib\Utility\file.llb\*Write To Spreadsheet File.vi*
0.0 ⇨ %	**EGU to %.vi**	*C:\LABVIEW\vi.lib\addons\PID\prctrl.llb\*EGU to %.vi*
load Control	**Load Fuzzy Controller.vi**	*C:\LABVIEW\vi.lib\addons\fuzzy.llb\*Load Fuzzy Controller.vi*
Fuzzy-Control	**Fuzzy Controller.vi**	*C:\LABVIEW\vi.lib\addons\fuzzy.llb\*Fuzzy Controller.vi*

C.2 List of subVIs for *I to v.vi and v to I.vi*:

Read from File	**Read from file.vi**	D:\mahesh\project\Data file*Read from file.vi*
Spline Inter-polant	**Spline Interpolant.vi**	*C:\LABVIEW\vi.lib\Analysis\5stat.llb\*Spline Interpolant.vi*
Spline Interp	**Spline Interpolation.vi**	*C:\LABVIEW\vi.lib\Analysis\5stat.llb\*Spline Interpolation.vi*

**available in function palette in Block Diagram.*

Appendix D

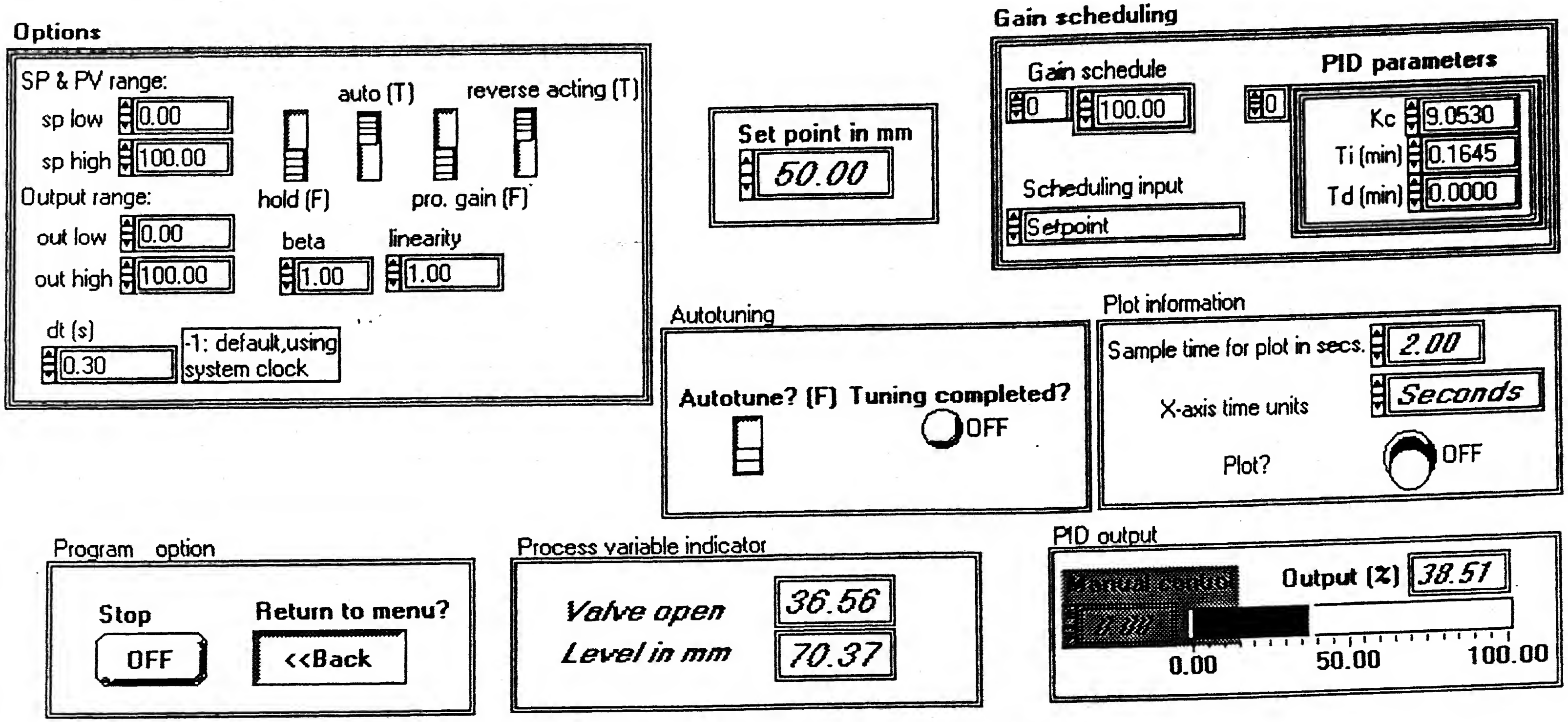
Read Me File

Refer to the Figure D.1, which is a Front Panel of VI: *Autotuning PID control of liquid level.* The file is located in *D:\mahesh\project\level control\ Autotuning PID control of liquid level.vi.* To start the program click the stop, Boolean Control to ON state in program option menu. User can fill in the parameters regardless of the Boolean Control state. After starting the program user can see the level and valve opening values in numeric indicators. Also user has option to plot graph, just click Plot? Boolean Control to ON state. Figure D.2 shows the plots. Figure D.1 and Figure D.2 are part of the same front panel. The 1st chart will show online data coming with respect to data points. 2nd and 3rd graphs will show up only after stopping the program by using Stop Boolean Control. There is also the option of saving data. After closing the program a interactive dialog box appears which guides the user to save the data. For autotuning just click the Boolean Control Autotune?. Then user can proceed according to instructions on the front panel of the autotuning wizard.

Refer to the Figure D.3, which is a Front Panel of VI: *Fuzzy Logic control of liquid level.* The other part of the front panel showing plots is same as Figure D.2. The file is located in *D:\mahesh\project\fuzzy\Fuzzy Logic control of liquid level.vi.* For fuzzy control user has to load the file *level_control.fc* file, which contains control *rule database* and is located in same file path. Plot function is same as above.

If program gives any errors or unable to control liquid level which can be seen from very erratic valve movements exceeding its saturation limits or valve doesn't moves at all, check the following points:

1. If there are any data acquisition errors, check whether chassis is powered ON. Before *opening LabVIEW* for running the program make sure that *chassis* is *powered ON first.*
2. Check whether *Driver Circuit* is working propely. Input supply for sensors should not be less than 5 VDC. Otherwise sensors will not give smooth voltage output. Also check the voltage supply to the motor of valve. It should be between 12 to 15 VDC.



sp low 0.00
sp high 100.00
out low 0.00
out high 100.00
beta 1.00
linearity 1.00
dt (s) 0.30
-1: default,using system clock

Set point in mm 50.00

Gain schedule 0 100.00
Scheduling input Setpoint
0
Kc 9.0530
Ti (min) 0.1645
Td (min) 0.0000

Autotune? (F)
Tuning completed? OFF

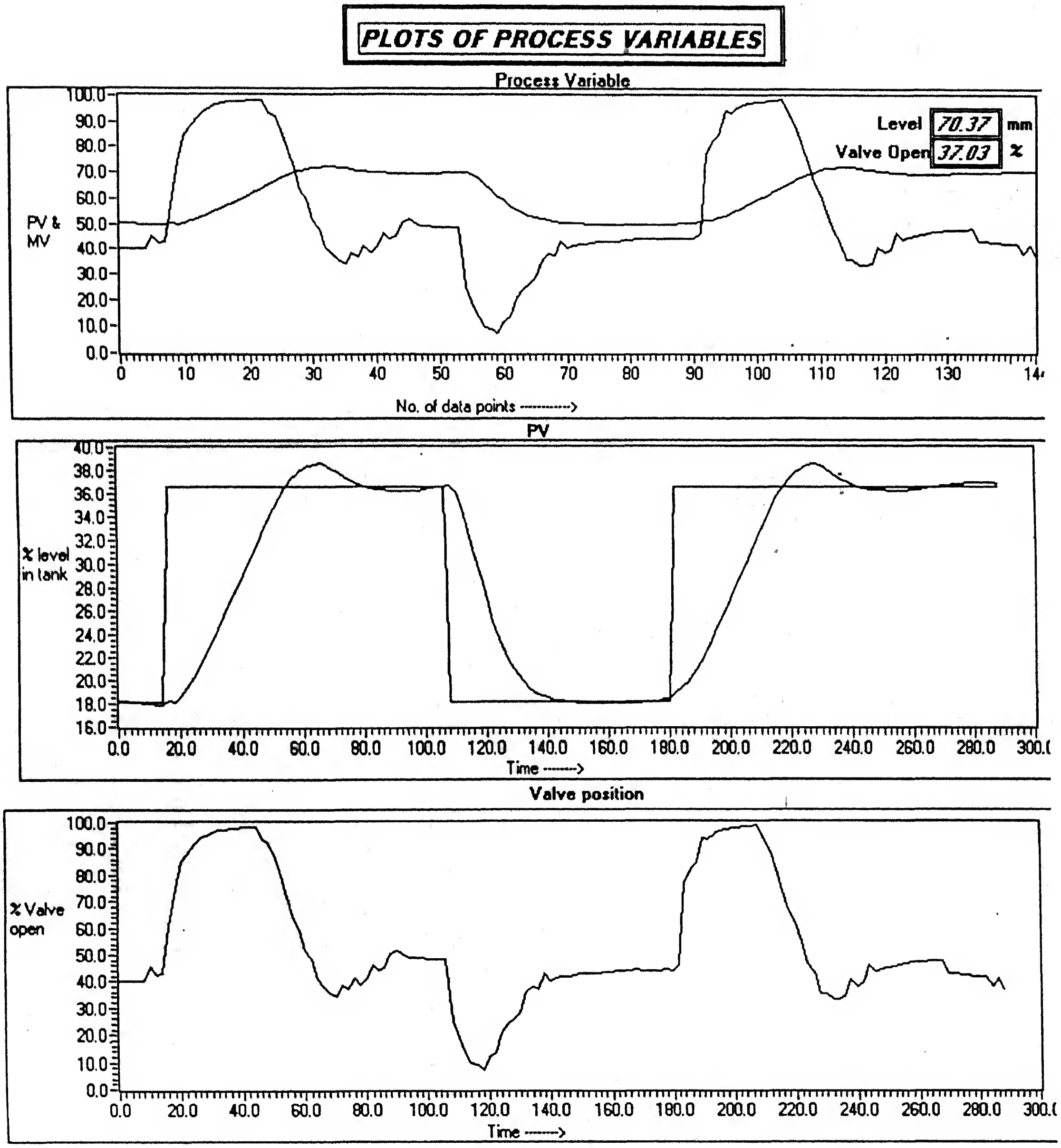
Sample time for plot in secs. 2.00
X-axis time units Seconds
Plot? OFF

Stop OFF
Return to menu? <<Back

Valve open 36.56
Level in mm 70.37

Output (%) 38.51

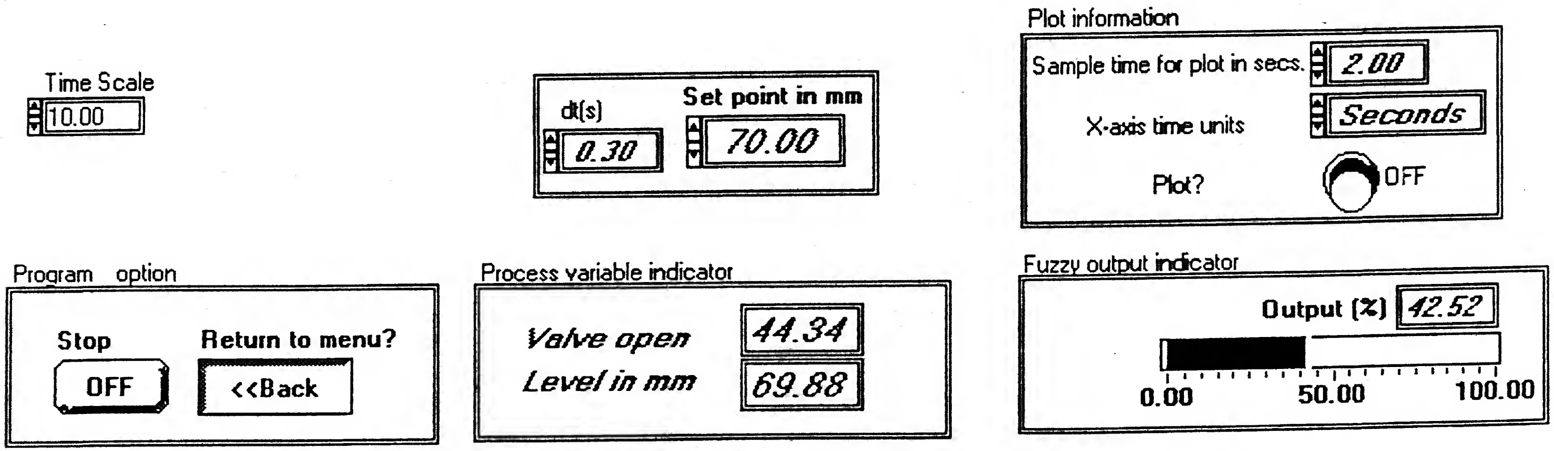
Figure D.1: Front Panel of VI: Autotuning control of liquid level



Level 70.37 mm

Valve Open 37.03 %

Figure D.2: Front panel showing plots of process variables.



Time Scale 10.00

dt(s) 0.30

Set point in mm 70.00

Plot information

Sample time for plot in secs. 2.00

X-axis time units Seconds

Plot? OFF

Program option

Stop OFF

Return to menu? <<Back

Process variable indicator

Valve open 44.34

Level in mm 69.88

Fuzzy output indicator

Output (%) 42.52

Figure D.3: Front panel of VI: Fuzzy logic control of liquid level.

Appendix E

Control Equations

E.1 PID controller model:

The PID VIs available in software package uses the positional PID control algorithm. The equations used in this algorithm are as follows.

PV filtering:

$$PV_f = 0.5*PV+0.25*PV(k-1)+0.175*PV*(k-2)+0.075*PV*(k-3) \qquad \text{(E.1)}$$

Error Calculation:

Integral and Derivative action:

$$e(k)=\left(SP-PV_f\right)*\left(L+(1-L)*\frac{\left|SP-PV_f\right|}{SP_{rng}}\right) \qquad \text{(E.2)}$$

Proportional Action:

$$eb(k)=\left(\beta*SP-PV_f\right)*\left(L+\left(1-L\right)*\frac{\left|\beta*SP-PV_f\right|}{SP_{rng}}\right) \qquad \text{(E.3)}$$

SP_{rng} is the range of the set point (set to 0 to 100 % of the process variable range).

β is the set point factor (ranges from 0 or 1)

L is the linearity factor (ranges from 0 to 1)

Proportional Action:

$$u_p=\left(k_c*eb(k)\right) \qquad \text{(E.4)}$$

Integral Action:

$$u_I(k)=\frac{k_c}{\tau_I}*\left(\sum_{i=1}^{k}\left[\frac{e(i)+e(i-1)}{2}\right]*\Delta t*\left[\frac{1}{1+\frac{10*e(i)^2}{SP_{rng}^2}}\right]\right) \qquad \text{(E.5)}$$

Derivative Action:

$$u_D(k)=(-k_c)*\frac{\tau_D}{\Delta t}*\left(PV_f(k)-PV_f(k-1)\right) \qquad \text{(E.6)}$$

Model of PID controller:

$$u(t)=k_c*\left[\left(\beta*SP-PV\right)+\frac{1}{\tau_I}\int_0^t\left(SP-PV\right)dt-\tau_D*\frac{dPV_f}{dt}\right] \qquad \text{(E.7)}$$

E.2 Zigler-Nichols settings for Autotuning under PI control:

Controller Type	Propotional Gain k_c	Integral Time τ_I	Derivative Time τ_D
Propotional only, P	$\frac{0.44\tau_P}{t_d}$	-	-
Propotional-Integral, PI	$\frac{0.4\tau_P}{t_d}$	$5.33t_d$	-
Propotional-Integral-Derivative, PID	$\frac{0.53\tau_P}{t_d}$	$4t_d$	$0.8t_d$

E.3 Cohen-Coon settings:

Controller Type	Propotional Gain k_c	Integral Time τ_I	Derivative Time τ_D
Propotional only, P	$k_c = \frac{1}{K}\frac{\tau_p}{t_d}\left(1+\frac{t_d}{3\tau_p}\right)$	-	-
Propotional-Integral, PI	$k_c = \frac{1}{K}\frac{\tau_p}{t_d}\left(0.9+\frac{t_d}{12\tau_p}\right)$	$\tau_I = t_d \frac{30+3t_d/\tau_p}{9+20t_d/\tau_p}$	-
Propotional-Integral-Derivative, PID	$k_c = \frac{1}{K}\frac{\tau_p}{t_d}\left(\frac{4}{3}+\frac{t_d}{4\tau_p}\right)$	$\tau_I = t_d \frac{32+6t_d/\tau_p}{13+8t_d/\tau_p}$	$\tau_D = t_d \frac{4}{11+2t_d/\tau_p}$

E.4 Filtering Equations:

The FIR Windowed filter uses convolution equations.

The convolution h(t), of the signals x(t) and y(t) is defined as

$$h(t) = x(t) * y(t) = \int_{-\infty}^{+\infty} x(\tau) y(t-\tau) d\tau$$

(E.8)

where the symbol * denotes convolution.

For the discrete implementation of the convolution, let h represent the output sequence X * Y, let n be the number of elements in the input sequence X, and let m be the number of elements in the input sequence Y. Assuming that indexed elements of X and Y that lie outside their range are zero,

$x_i = 0, i < 0$ or $i \geq n$

and

$y_j = 0$, j < 0 or $j \geq m$

then elements of h are obtained using,

$$h_i = \sum_{k=0}^{n-1} x_k y_{i-k} \tag{E.9}$$

for i = 0, 1, 2, ..., size-1,

size = $n + m$ - 1,

where size denotes the total number of elements in the output sequence X * Y.

References

Astrom K.J. and T. Hagglund (1984). Autotuning of simple regulators with specification on phase and amplitude margins. Automatica, 20(5), 645-651.

Astrom K. J., J. J. Anton, and K.E. Arzen (1986). Expert Control. Automatica, 27(4), 599-609.

Astrom K. J. and B. Wittenmark (1995). Adaptive Control. 2nd edition, Addison-Wesley Publishing Company, Inc.

Braae M. and D. A. Rutherford (1979). Theoretical and Linguistic Aspects of the fuzzy logic controller. Automatica, 15(5), 553-577.

Braae M. and D. A. Rutherford (1979). Selection of Parameters for fuzzy logic controller. Fuzzy sets and systems, 2, 185-199.

Bezdeck James C. (1993). Fuzzy Models - What are they and why? IEEE Trans. on fuzzy systems. 1(1), 1-6.

Bristol E. H. (1973). Adaptive Process control: Versatile online tool. Control Engg., 20(4), 41-47.

Czogala E. and W. Pedrycz (1981). An identification of fuzzy systems and it's application in control problems. Fuzzy sets and systems, 6, 73-83.

Czogala E. and W. Pedrycz (1982). Control problems in fuzzy systems. Fuzzy sets and systems, 7(3), 257-273.

Doebelin Ernest O. (1975). Measurement Systems. Revised edition, McGraw-Hill Kogakusha.

Didier, Dubois and Henri Prade (1987). Mean value of fuzzy number. Fuzzy set and systems, 24(3), 279-300.

Dimiter Driankov, Hans Hellendroom, Michael Reinfrank (1993). An Introduction to Fuzzy Control, Narosa Publishing House.

Engell S. and T. Heckentheller (1995). Fuzzy Control-an Alternative to Model Based Control?, R. Berber (ed.), Methods of Model Based Process Control, Kluwer Academic Publishers, 755-773.

Graham Bruce P. and Robert B. Newell (1988). Fuzzy identification and control of liquid level rig. 26, 255-273.

Gupta S. and J. P. Gupta (1989). PC interfacing for Laboratory Data Acquisition and Process Control. ISA, Research Triangle Park.

Hagglund T. and K. J. Astrom (1991). Industrial adaptive controllers based on frequency response techniques. Automatica, 27(4), 599-609.

Ibrahim E., R. Cockrum, G. Herder and F. Smith (1993). Integrating Computers and Industrial Hardware in Instrumentation and Control Education, ASEE Annual Conference Proceedings, 1425-1433.

Kauler B. (1993). Dataflow and Visual Programming the Way Ahead for Engineers.

Kickert W. J. M. and H. R. Van Nauta Lemke (1976). Application of a fuzzy controller in a warm water plant. Automatica, 12(4), 301-338.

King P.J. and E. H. Mamdani (1977). The application of fuzzy control systems to Industrial processes. Automatica, 13(3), 235-242.

McMillan G. K. (1986). Advanced Algorithms: Beware of false prophecies. 1, Intech, 55-57.

Mamdani E. H. and S. Assilian (1975). An experiment in Linguistic Synthesis with fuzzy logic controller. Int. J. of Man-Machine studies, 7, 1-13.

Remberg C., G. Field, G. Wozny and F. N. Fett (1995). R. Berber (ed.), Methods of Model Based Process Control, Kluwer Academic Publishers, 785-796.

Seborg D. E., T. F. Edger and S. L. Shah (1986). Adaptive control strategies for process control: A survey. AIChE J., 32, 881-913.

Shinskey F. G. (1979). Process-Control Systems. 2nd edition, McGraw-Hill.

Smith Carlos A. and Corripio Armando B. (1997). Principles and Practice of Automatic Process Control. 2nd edition, John Wiley.

Sugeno M. (1985). An Introductory Survey of Fuzzy Control, Information Sciences, 36, 59-83.

Takagi T. and M. Sugeno (1985). Fuzzy Identification of Systems and Its Applications to Modeling and Control, IEEE Trans. on Systems, Man, and Cybernetics, 15(1), 116-132.

Tong R. M., M. B. Beck and A Latten (1980). Fuzzy control of the Activated Sludge Wastewater Treatement process. Automatica, 16(6), 659-701.

Tong R. M. (1980). The evaluation of fuzzy models derived from experimental Data. Fuzzy sets and systems, 4, 1-12.

Zadeh L. (1965). Fuzzy Sets. Information and control, 8, 338-353.

Zadeh L. (1968). Fuzzy Algorithms. Information and control, 12, 94-102.

Zadeh L. (1973). Outline of a new approach to the analysis of complex systems. IEEE Trans. on System, Man and Cybernetics, 3(1), 28-44.

Zimmerman H. J. (1987). Fuzzy Sets, Decision Making and Expert Systems. Kluwer Academic Publishers.